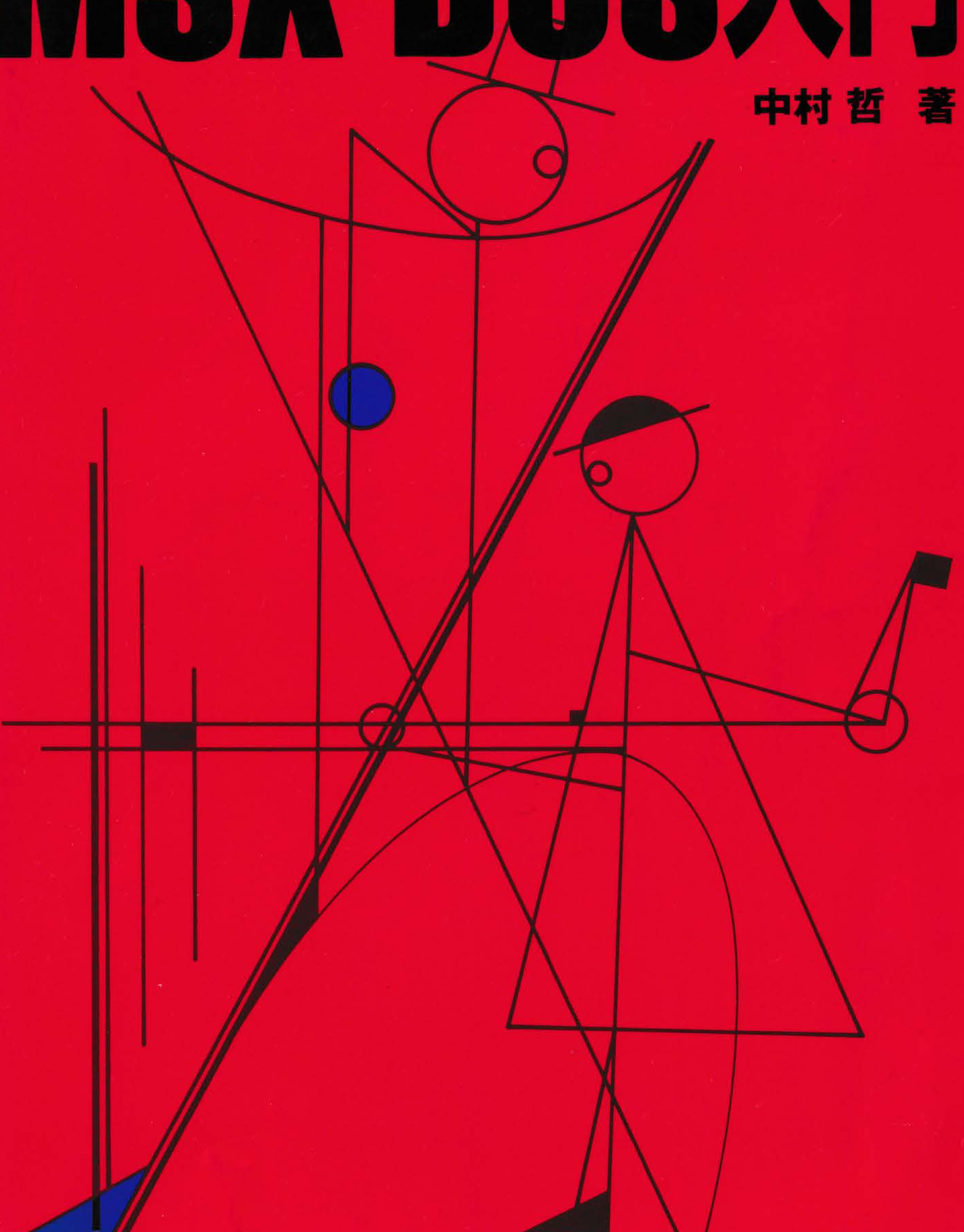
MSX-DOS入門

中村哲 著



アスキー出版局

Designed by STARKA
Illustrated by Mari Kobayashi

MSX-DOS入門

中村　哲　著

アスキー出版局

商標
CP／M は米 Digital Research 社の登録商標です.
MS-DOS は米 Microsoft 社の登録商標です.
MSX，MSX2，MSX-DOS は株式会社アスキーの商標です.
UNIX オペレーティング・システムは，AT&T ベル研究所が開発し，AT&T がライセンスしています.

はじめに

一昔まえまで，コンピュータは扱いが難しく，しかも高価なものであり，一般人には縁のないものと考えられがちでした．しかし，いまや小学生からお年寄りまでが，ごく普通にコンピュータを使っています．MSX の普及により，コンピュータもようやく家電製品なみの手軽さで取り扱うことができるようになりました．

コンピュータの家庭への浸透は，まず電子レンジ，クーラー，ビデオなどの家電製品に組み込まれることによって，知らず知らずのうちに始まりました．これをコンピュータの第1次普及としますと，第2次普及はなんといっても家庭用ゲーム機によるものです．MSX をゲーム・マシンのつもりで購入した人も多いのではないでしょうか？

しかし，MSX は単なるゲーム・マシンではなく，ホーム・コンピュータとしての本格的な使用に耐えうる性能を持っています．そのため，最近では MSX をよりコンピュータらしく使おうとしている人も増えてきているようです．この傾向に拍車をかけているのが，最近のディスク・ドライブの普及です．

コンピュータの本格的な使用にディスク・ドライブは必需品ですが，高価なことが最大の欠点でした．しかし，最近のエレクトロ技術の進歩のためディスク・ドライブも安価になり，MSX でも容易にディスク・ドライブを使用することができるようになりました．このためディスク・ドライブはかなり普及してきているようですが，そのディスク・ドライブをコントロールする DOS については，まだ，だれもが使いこなしているとはいえないのが現状です．中には，あまりにも急速にディスク・ドライブが普及したため，DOS の必要性，有用性を知らずに，ディスク・ドライブを使っている人も多いようです．

本書は，MSX-DOS の使用法を説明することによって，DOS の必要性，有用性について理解していただくことを目的としています．本書によって，1人でも多くの MSX ユーザーの皆さんが MSX-DOS を使いこなし，コンピュータ・ライフをより豊かなものとすることを願っております．

1988年1月　中村　哲

本書で使用したハードウェア

本書では以下のようなシステムを使用しています．ただし，本文中のサンプル画面については，MSX2を使用して表示文字幅を80文字に設定してあります．また，機種によりメッセージなどが異なる場合のサンプルについては，FS5500のものを使用しています．

● MSX本体＋ディスク・ドライブ

・2ドライブ内蔵型MSX2　(National FS5500)

・1ドライブ内蔵型MSX2　(MITSUBISHI ML-G30 model 1)

・ドライブ未装備型MSX　(TOSHIBA PASOPIA IQ HX-31)

ディスク・インターフェース・カートリッジ　(YAMAHA FD-051)

ディスク・ドライブ・ユニット　(YAMAHA FD-05J)

●ディスプレイ

●プリンタ

目　次

3章 ファイルのいろいろな処理 53
— ディスク上のファイルに対するコマンド操作 —

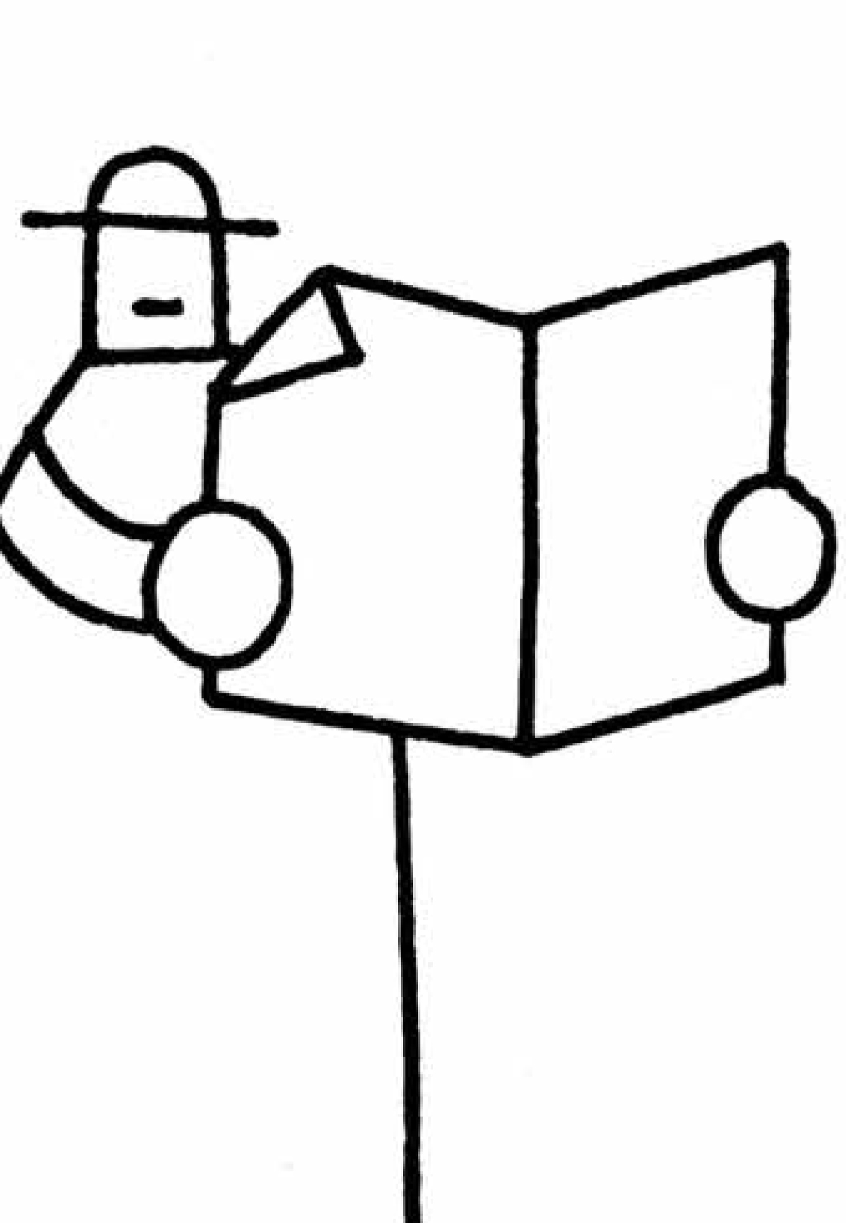
4章 周辺装置とのデータのやりとり 91
— デバイス・ファイルを使う —

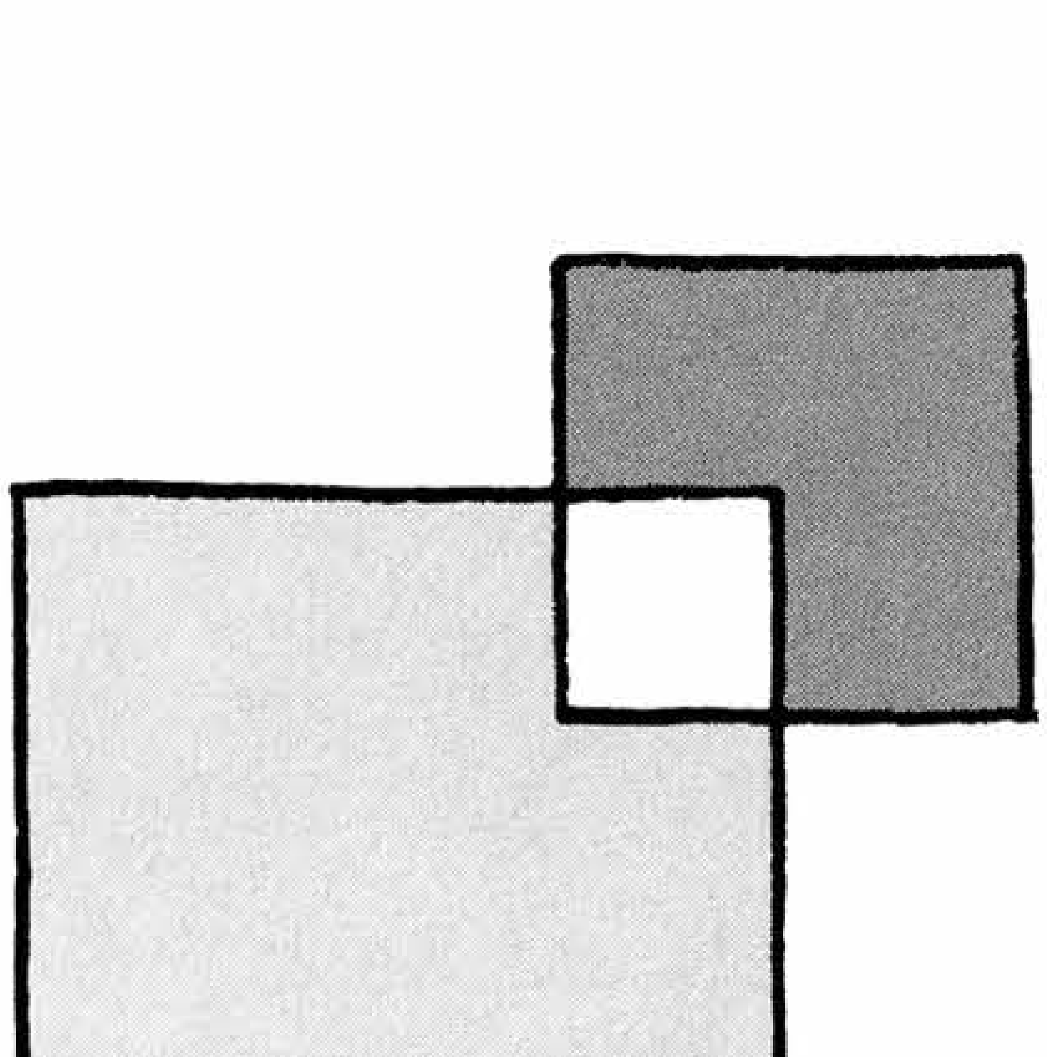
本文イラスト■細田雅亮

1章

MSXとMSX-DOS

―MSX-DOSの役割とは―



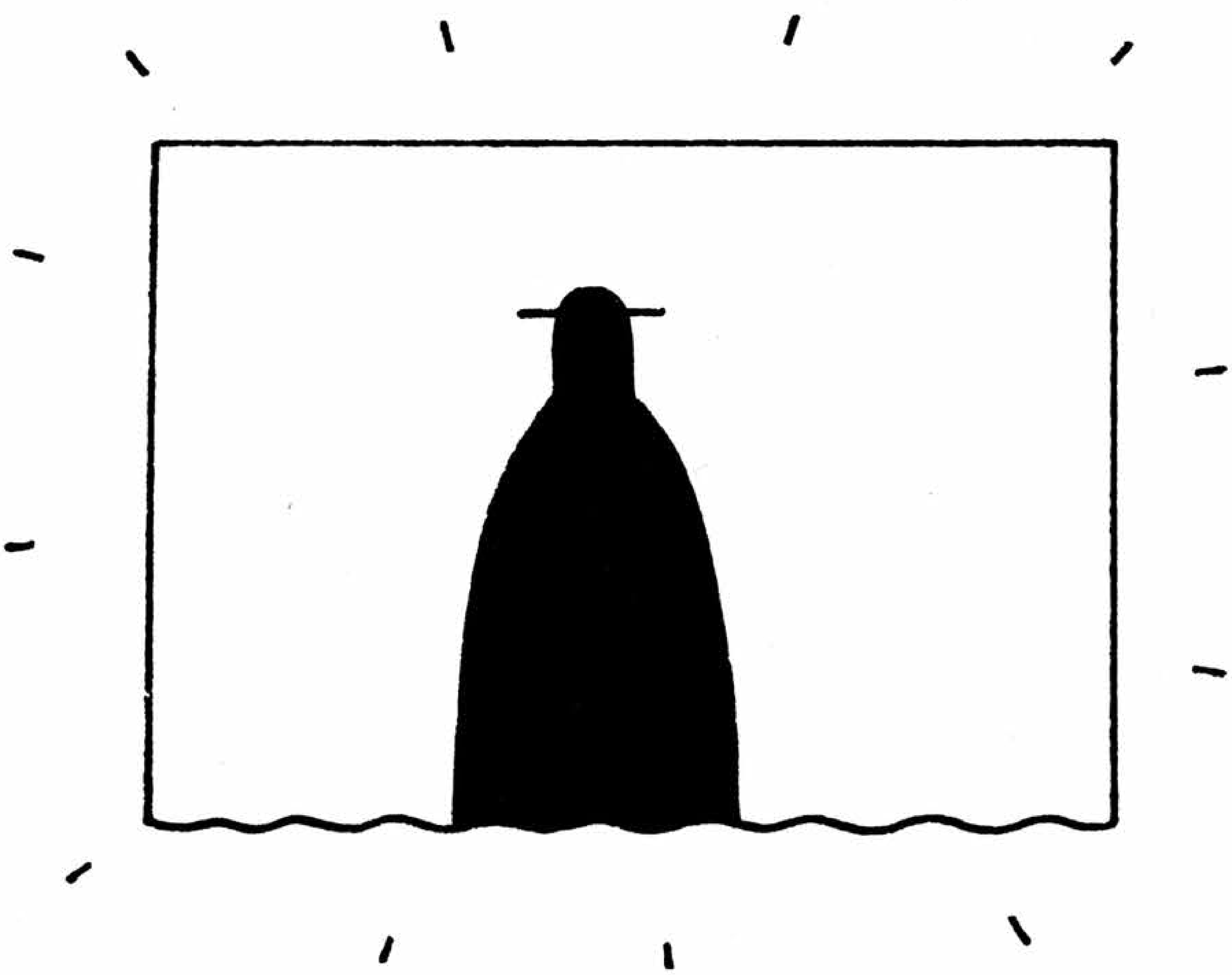


1-1 MSX-DOSとは何か?

MSX用のディスク・ドライブ・ユニットや，ディスク内蔵型のMSXを買うと，機種により，ラベルに"**MSX-DOS**"と書かれたディスクが添付されてくる場合があります．また，最近では，ディスクによって供給されるソフトウェアも増えてきていますが，それらのソフトウェアの中に，よく次のように解説されているものを見かけます．

「この製品はMSX-DOS上で動作します．」
「作成したデータは，MSX-DOSのファイルとして保存されます．」

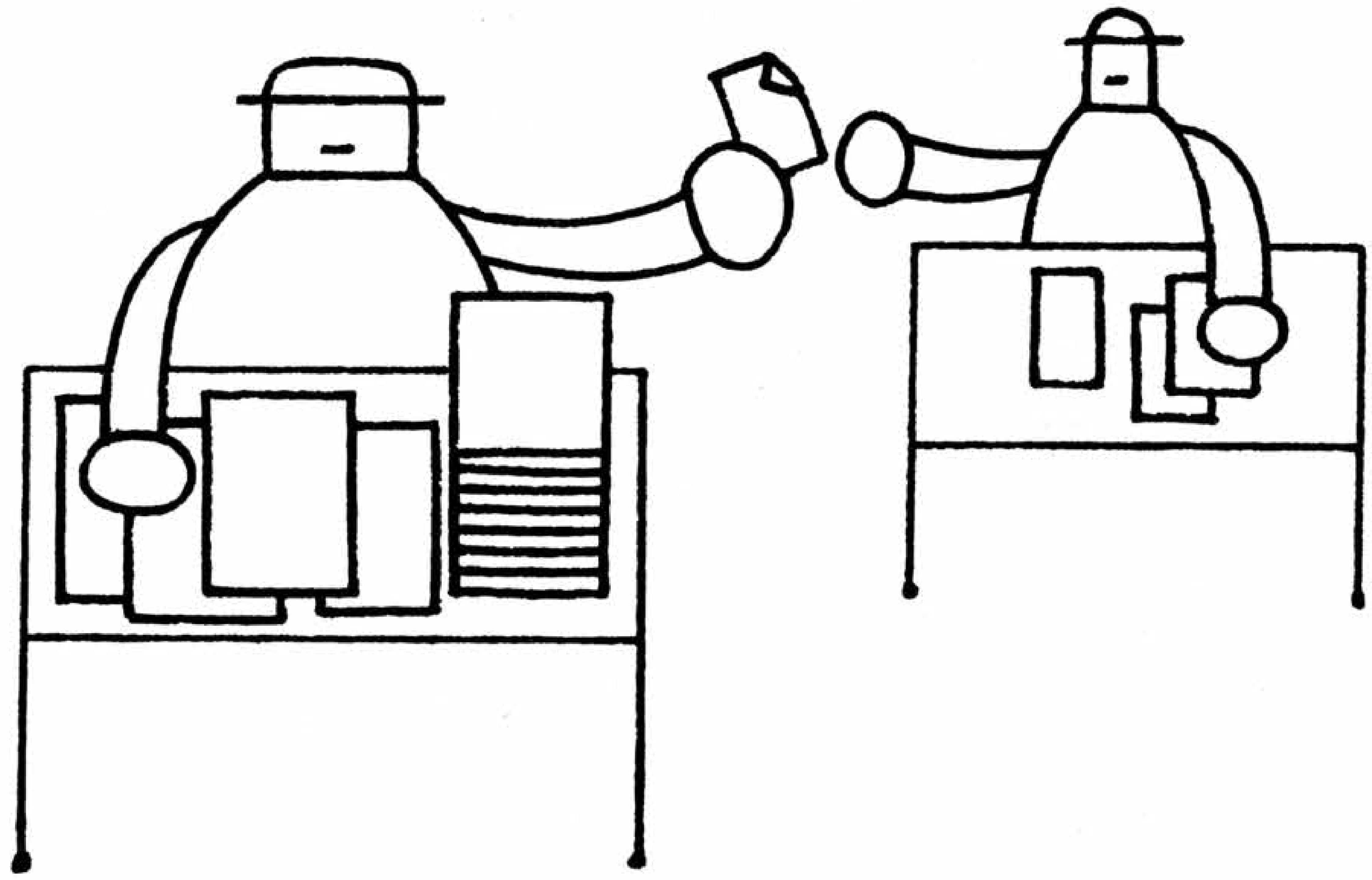
この，"MSX-DOS"とはいったいなんでしょうか？　本章では，MSX-DOSとは何か，どのように役立つのか，そして実際使用するにあたって，何を揃えればよいのか解説していきたいと思います．



ご存じのように，コンピュータは何をするにもプログラムが必要です．もし，ディスク・ドライブを使ってディスクに**アクセス**(データを書き込んだり，読み出したりすること)するなら，そのためのプログラムが必要になります．ディスク・ドライブを制御するプログラムとして，まず思い付くのは**DISK-BASIC**です．DISK-BASICを使えば，とりあえずディスクからプログラムやデータを読み込んだり書き込んだりできます．しかし，これではカセット・テープレコーダの使い方と変わりありません．ディスクを，単に速いカセット・テープとしてプログラムを保存するためだけに使うならば，DISK-BASICで十分でしょう．しかし，これでは，ディスク・ドライブの利点が十分に活かされていません．いくら値段が安くなってきたからといっても，ディスクをカセット・テープ代わりに使うのではもったいない話です．せっかくディスク・ドライブを装備しているのですから，大量のデータ，プログラムを高速に取り扱うなど，その性能をフルに活かせないものでしょうか．こうなるとDISK-BASICでは少々役不足です．ディスク上のデータ，プログラムをきちんと管理し，それらを即座に読み込み処理，実行するなどということはDISK-BASICにはできません．これは，何もBASICが悪いというのではありません．本来プログラムを記述するための言語であるBASICに，ディスク・ドライブを制御する機能まで持たせようとしたところに無理があるのです．

もし，ディスク・ドライブを制御するためだけのプログラムがあれば，このプログラムはディスク・ドライブの制御のみに専念すればよいので，BASICとは違った環境から利用することもできるはずです．このディスク・ドライブを制御するために生まれたプログラムが**DOS**(**Disk Operating System**)なのです．

DOSは内部に，ディスクの読み書きを行うための基本的なプログラムを持っているので，ほかのプログラム(BASIC，ワープロ，データベースなど)でディスクをアクセスしようと思ったら，細かい操作はDOSに下請けに出せばよいのです．つまり，DOSの存在によってディスク・アクセスのためのプログラムを個々のアプリケーション・プログラムが持っている必要はなくなります．



さらに，ディスク・アクセス以外にも，多くのプログラムで使う基本的な部分をDOSに含めてしまえば，個々のプログラムの負担はさらに減ります．この考えを押し進めれば，キーボードやディスプレイ，プリンタなど，他の周辺装置の制御もDOSの管理下に置くことができます．

DOSを使うことによって個々のプログラムは，周辺機器の制御などの基本的な処理を省略できるので，負担が減り，より高度で複雑な処理に専念できるのです．

しかし，DOSのメリットはそれだけではありません．すべてのプログラムがDOSという共通のルーチンを使うことにより，プログラム間の互換性が保たれるのです．具体的な例をあげてみましょう．

もしも，ワープロと，そのワープロで作った文章を加工するプログラム(文章に図版を差し込んだり，プリンタに出力する字体を変えるプログラムなど)が，まったく別の書式でディスクにデータを書き込んでいたらどうなるでしょうか？　ワープロで作った文章を，ほかのプログラムからは読めなくなってしまい，なんの意味もありません．しかし，実際にはDOS上のすべてのプログラムはDOSのディスク制御機能を使っているので，このような心配はおきません．

さらに，DOS上にさまざまな言語やアプリケーション・ソフトウェアを構築しておけば，必要になったときにディスクからこれらのプログラムを読み込めばよいので，常にコンピュータ本体内にプログラムを置いておく必要はありません．こうすれば，メモリも有効に使えますし，操作性も格段に上がるはずです．また，これにより，たとえBASIC以外の言語でも，好きなときにディスクから読み込んできて使うことができるのです．何もROM上に言語を固定しておく必要はありません．

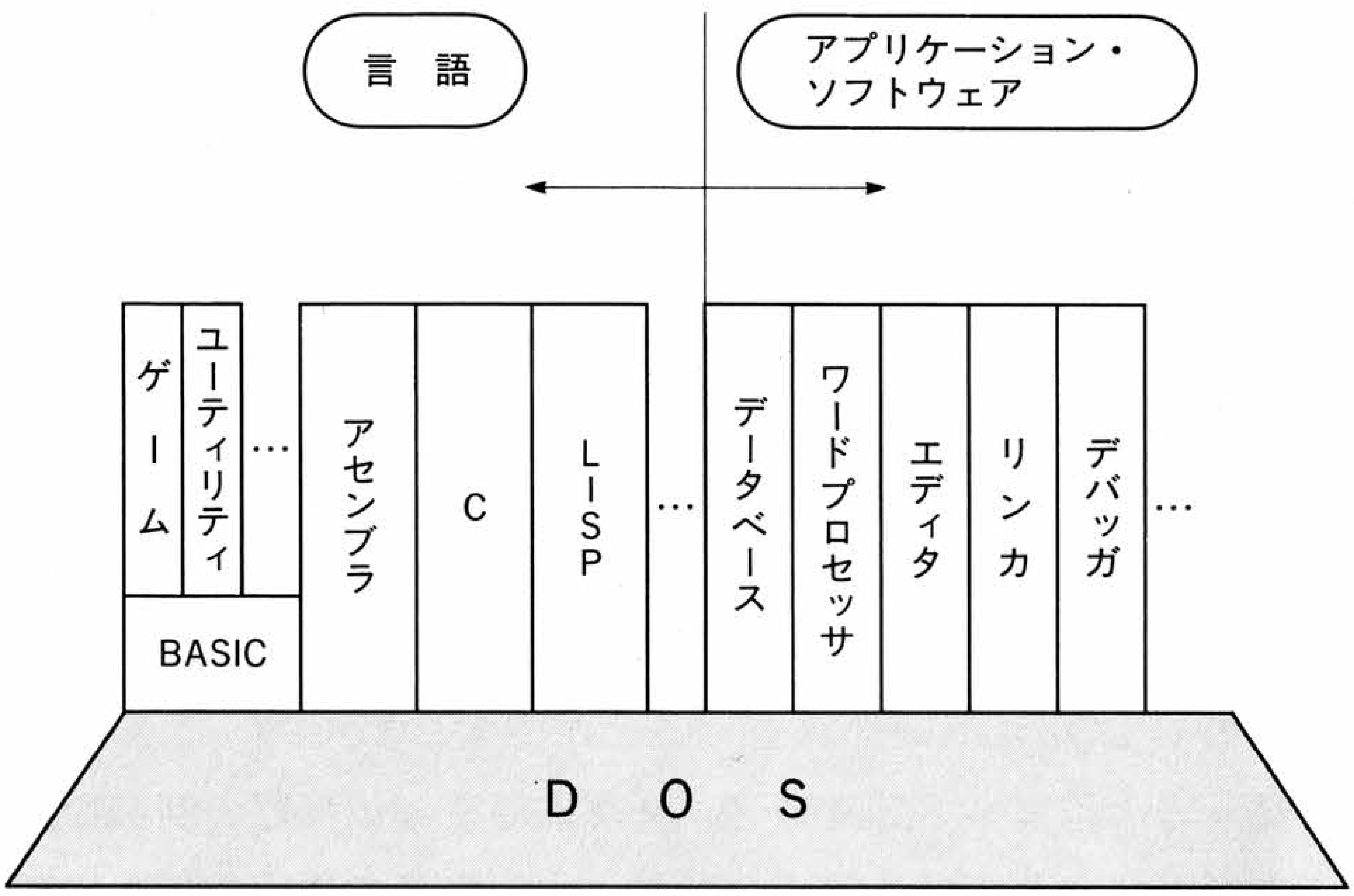


図1.1　DOSの上に構築されるソフトウェアの図

しかも，DOSには他機種との互換性という点でも大きなメリットがあります．MSXのように最初から互換性を考えて作られたコンピュータではあまり考える必要がないかも知れませんが，DOSが普及していない時代には，A社のコンピュータとB社のコンピュータでは同じCPU(中央処理装置)を使っているのにソフトウェアを共有できないということが当たりまえでした．

ところが，同じDOSさえ使っていれば同じソフトウェアを使えたので，DOSはソフトウェアの流通にも大いに役立ちました．他機種と同じソフトウェアが使えるということだけでも，DOSを使う意義があるのです．

では，次節では具体的にMSX-DOSの役割について見ていきましょう．

1-2 MSX-DOSのさまざまな役割

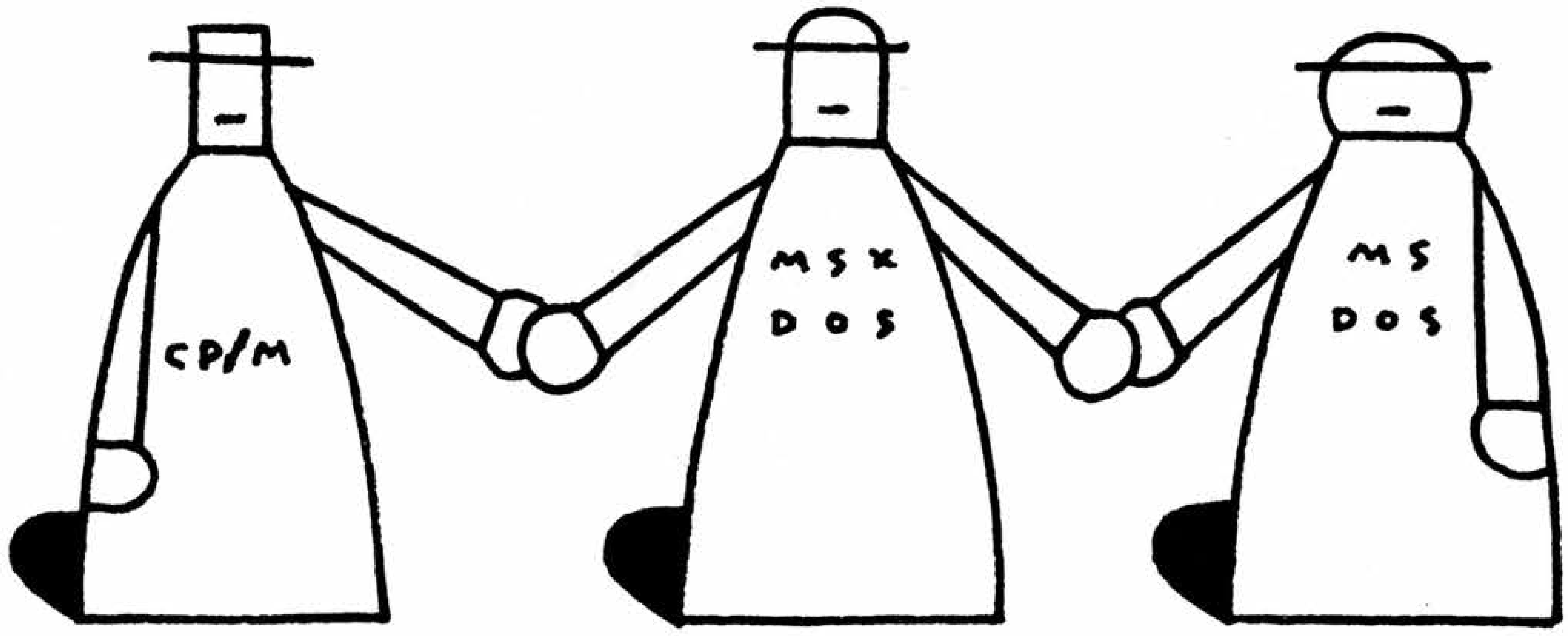
ここではMSX-DOSの役割というものを，他のDOSとの関係や位置づけという観点からさぐっていきたいと思います．

マイクロコンピュータのDOSを語っていくうえで，**CP／M**と**MS-DOS**の存在を避けては通れません．数年まえまで，8ビットのコンピュータの世界ではCP／MというDOSが圧倒的なシェアを誇っていました．機種が違っていても，CP／Mさえ動いていれば同じソフトウェアが使えるので，CP／M上のソフトウェアは大量に作られました．MSX-DOSは，このCP／Mとの互換性が十分に考えられているために，CP／M上で作成された豊富なソフトウェアを容易に移植することが可能となっています．

しかし，CP／MはマイクロコンピュータのDOSとしては初期のもので，いろいろと機能の限られたものでした．マイクロコンピュータの性能がアップし16ビットのコンピュータが登場してくると，MS-DOSという16ビット機用のDOSが主流となってきました．MSXは8ビットのコンピュータですが，将来16ビットのコンピュータを使う際にMS-DOSに移行しやすいように，MSX-DOSとMS-DOSの操作法は似ています．具体的に説明しましょう．DOSを利用してディスクを管理できるのは，何もプログラム中からだけではありません．われわれが，キーボードからDOSに直接命令を与えることによりDOSを操作することができます．この直接DOSに与える命令を**コマンド**と呼びます．MSX-DOSとMS-DOSでは，このコマンドの名前と機能が共通になっています．

さらに，8章でも詳しく述べますが，MS-DOSとMSX-DOSの間では，ディスクを介して，**ファイル**(DOSで扱うあるデータのかたまり)のやり取りも可能です．

このようにMSX-DOSは，CP／MとMS-DOSの架け橋としての役割も持っています．



では，ここでMSX-DOSの特徴を箇条書きにしてみましょう．

① MSX-DOSはディスクや周辺装置の細かい管理を受け持ってくれます．したがってユーザーはMSX-DOSが具体的にどのような処理を行っているかを気にすることなしに，MSX-DOS上のソフトウェアを使用することができます．
② MS-DOSと共通のコマンドを使って簡単にファイルの整理が行えます．
③ CP／Mの豊富なソフトウェアを容易に移植することが可能です．
④ MS-DOSとファイルを相互に読み書きできます．

MSX-DOSの利点は実際に使っていれば，おのずとわかるでしょうから，とりあえず使ってみることにしましょう．

1 3 MSX-DOSシステムの構築

MSX-DOS を実際に使うには，次のようなハードウェアが必要です．

①64K バイト以上のメイン RAM を実装した MSX，もしくは MSX2
②ディスプレイ
③フロッピー・ディスク・ドライブを 1 台以上

①と②はいままでに，MSX を使っていた人なら持っているはずです．もし，メイン RAM が 64K バイト未満の MSX を使用している人は RAM を増設して 64K バイト以上にしてください．また，以下ではとくに明記した場合以外は MSX2 と MSX を区別せず，両者を含めた広い意味で MSX という言葉を使うことにします．

③のフロッピー・ディスク・ドライブは，DOS を起動させるのですから必需品です．2 台以上あれば便利ですが，1 台でもかまいません．

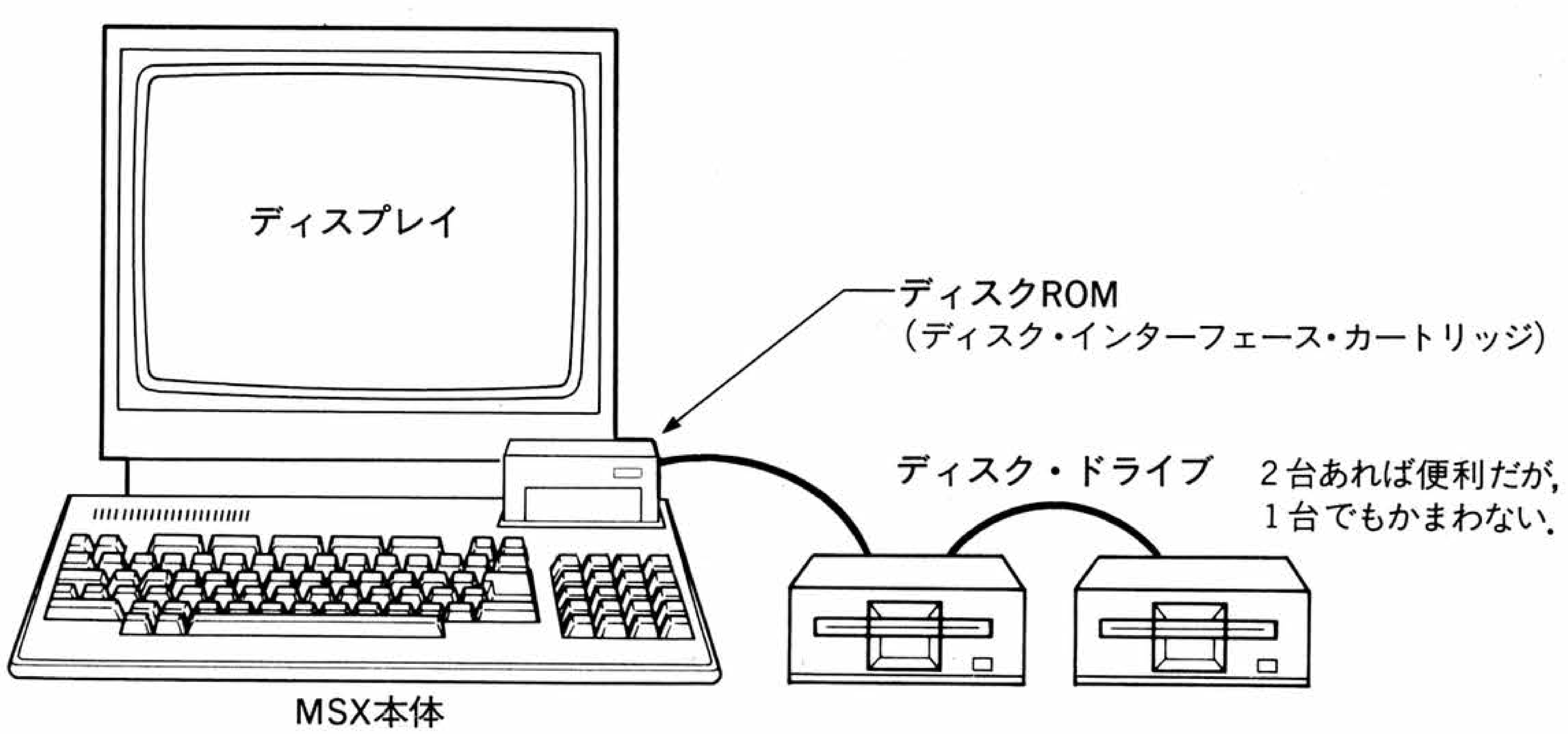


図 1.2　MSX-DOS に必要なハードウェア機器構成

次に必要なソフトウェアですが，MSX-DOSが機能するうえで，最低この3つのソフトウェアを欠くことはできません.

①ディスクROM
② MSXDOS.SYS
③ COMMAND.COM

①のディスクROMは，ディスク・インターフェイス・カートリッジの中に含まれているので，ディスク・ドライブを用意すればとくに気にする必要はありません．もちろんディスク内蔵型のMSXには，ディスクROMも内蔵されています.

②と③はMSX-DOSを構成する基本的なファイルで，MSX-DOSの**システム・ディスク**（**写真1.1**）に記録されています.



写真1.1　MSX-DOS　システム・ディスク

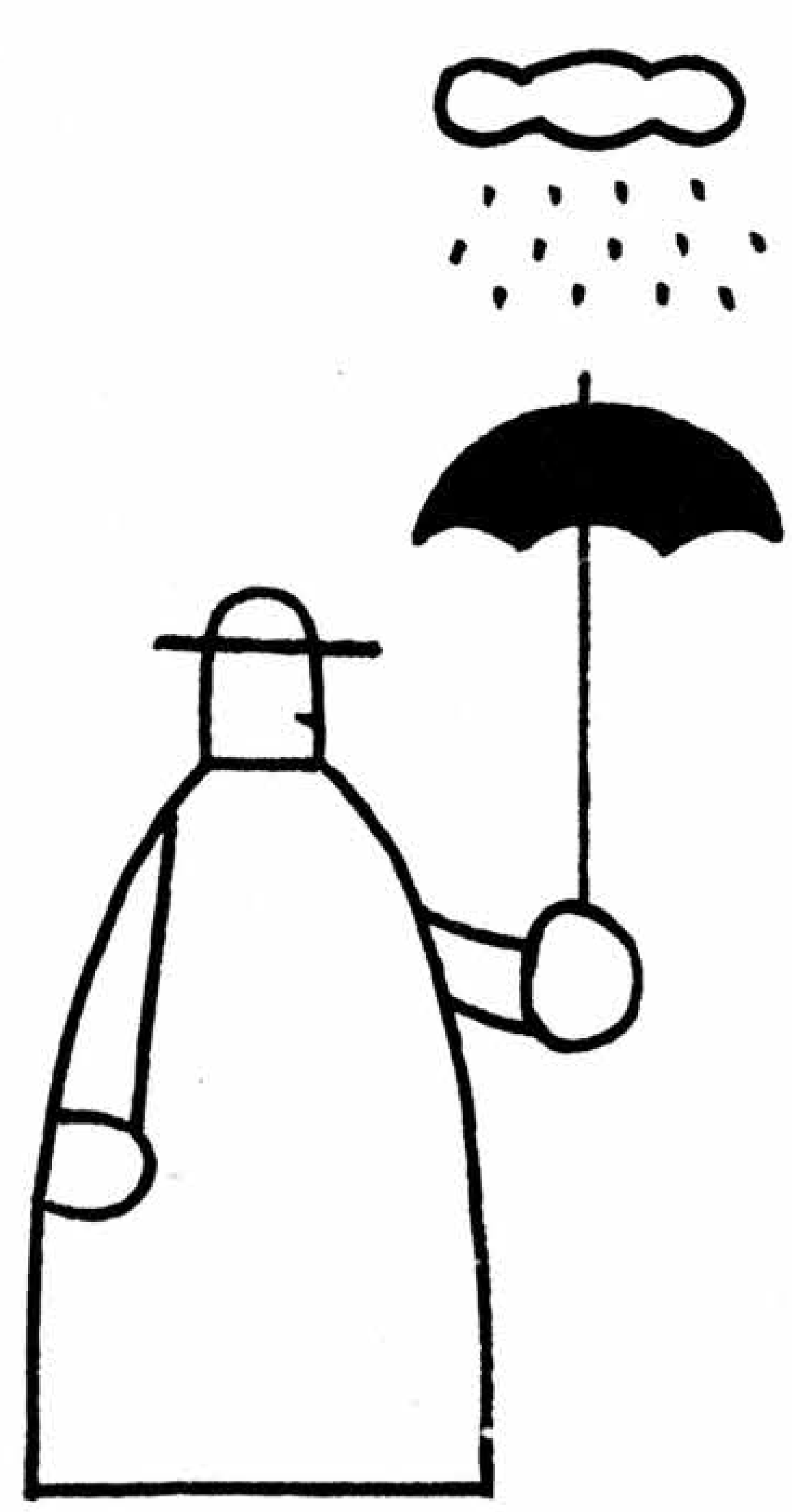
MSX-DOSのシステム・ディスクについては，MSX用のディスク・ドライブ・ユニットや，ディスク内蔵型のMSX購入時に添付されているのが普通ですが，機種によっては添付されていないものもあるので，その場合は，先の2つのファイルを含んだアプリケーション・ソフトウェア（「MSX-DOS TOOLS」など）を用意する必要があります.

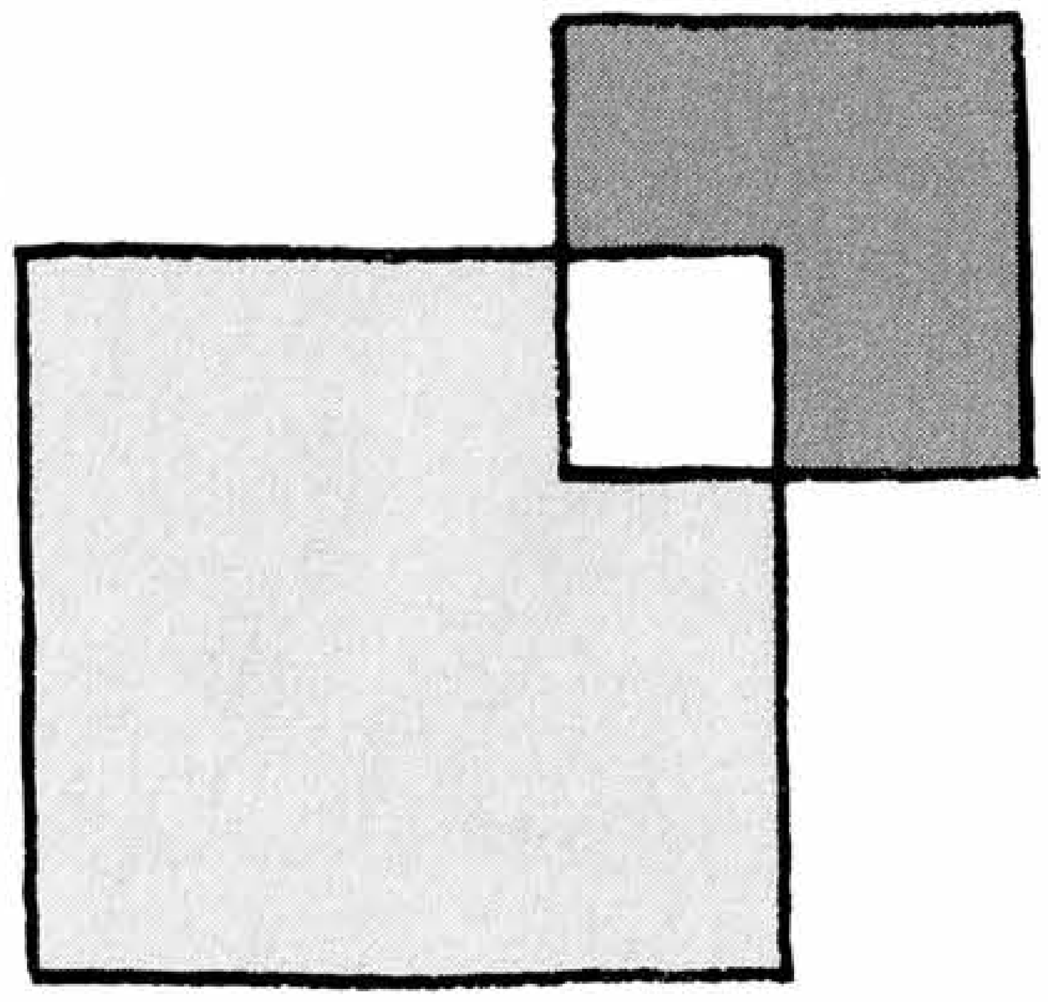
これらの準備が整ったら，いよいよ2章では，実際にMSX-DOSを起動させてみましょう.

2章

MSX-DOSの基本動作

―基本操作を覚える―





2-1 MSX-DOSの起動

これから, MSX-DOS を実際に起動させますが, そのまえにディスク・ドライブの名前について少し説明しておきましょう.

MSX-DOS では, ディスク・ドライブを 8 台まで使用することができます. そのため, それぞれのドライブをアルファベット 1 文字を使って表し, 区別しています. つまり, ドライブ A, ドライブ B, ドライブ C, ……, ドライブ H といった具合です.

ドライブ A というのは, MSX-DOS を起動するために必要なドライブで, 具体的には次のようなドライブを指します.

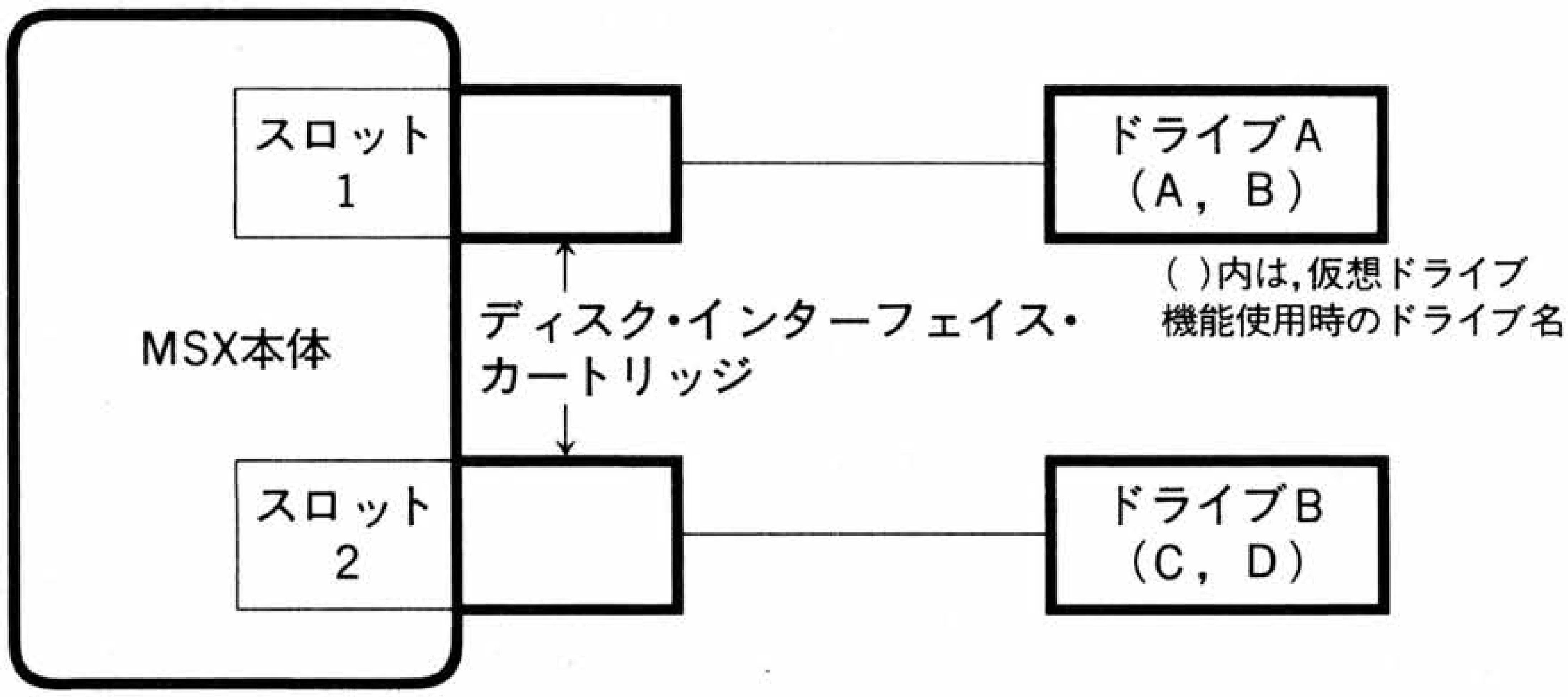
①接続, もしくは内蔵されているディスク・ドライブが 1 台だけならば, それがドライブAです.

②内蔵型ドライブを 2 台持っている MSX の場合は, "A"もしくは, "1"と記されたドライブのことです.

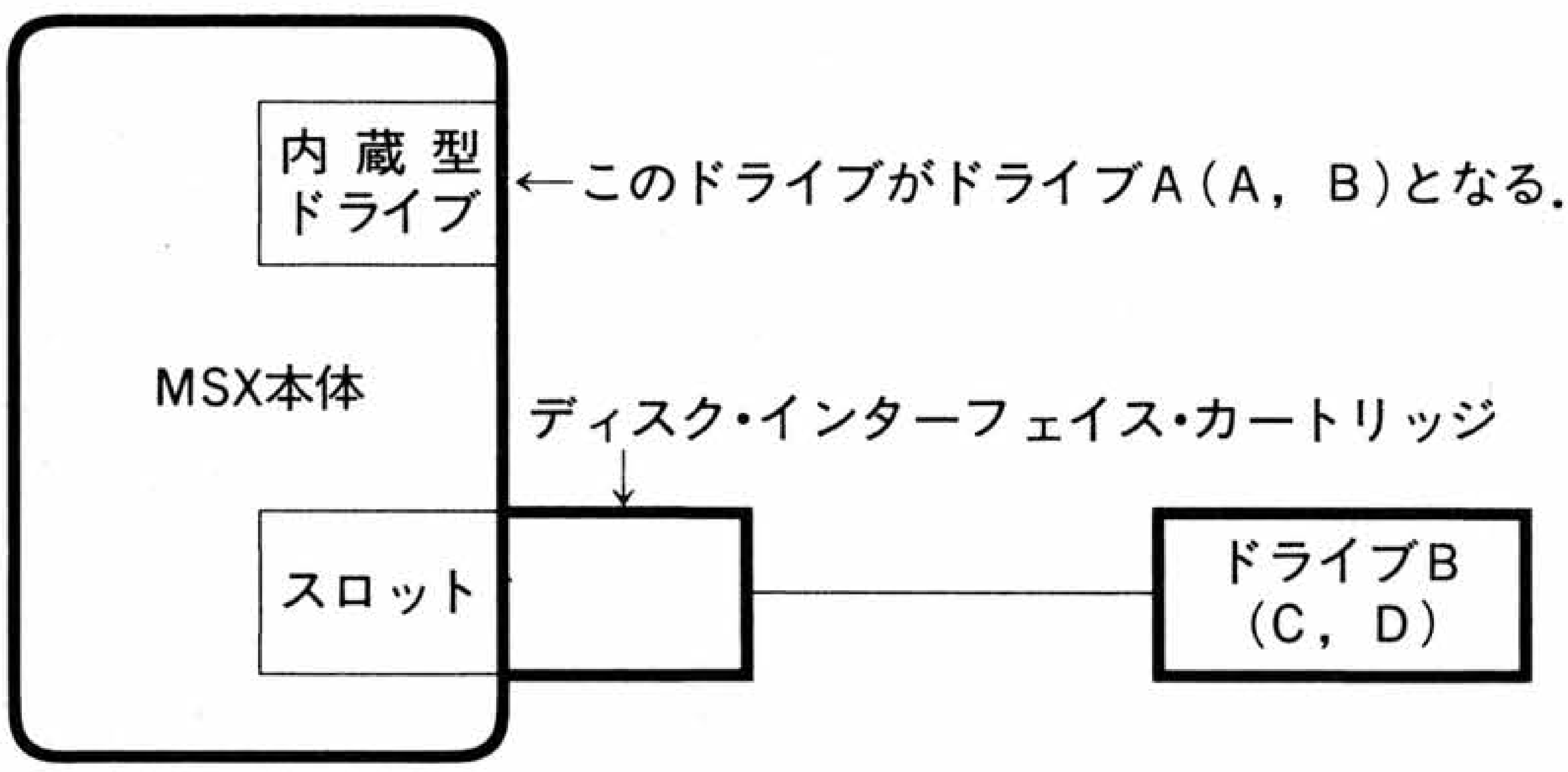
③内蔵ドライブを持たない MSX にディスク・ドライブを接続している場合は, スロット 1 のディスク・インターフェイス・カートリッジに直接つながっているドライブがドライブAです.

ドライブ A についてはこれでわかりますが, ドライブ B, C などについては, あなたの MSX の構成によってどのドライブを指すかが変わってきます. **図 2.1** を見てください. ただし, ドライブが 1 台の場合は, そのドライブがドライブ A とドライブ B の役割を果たします(詳しくは後述).

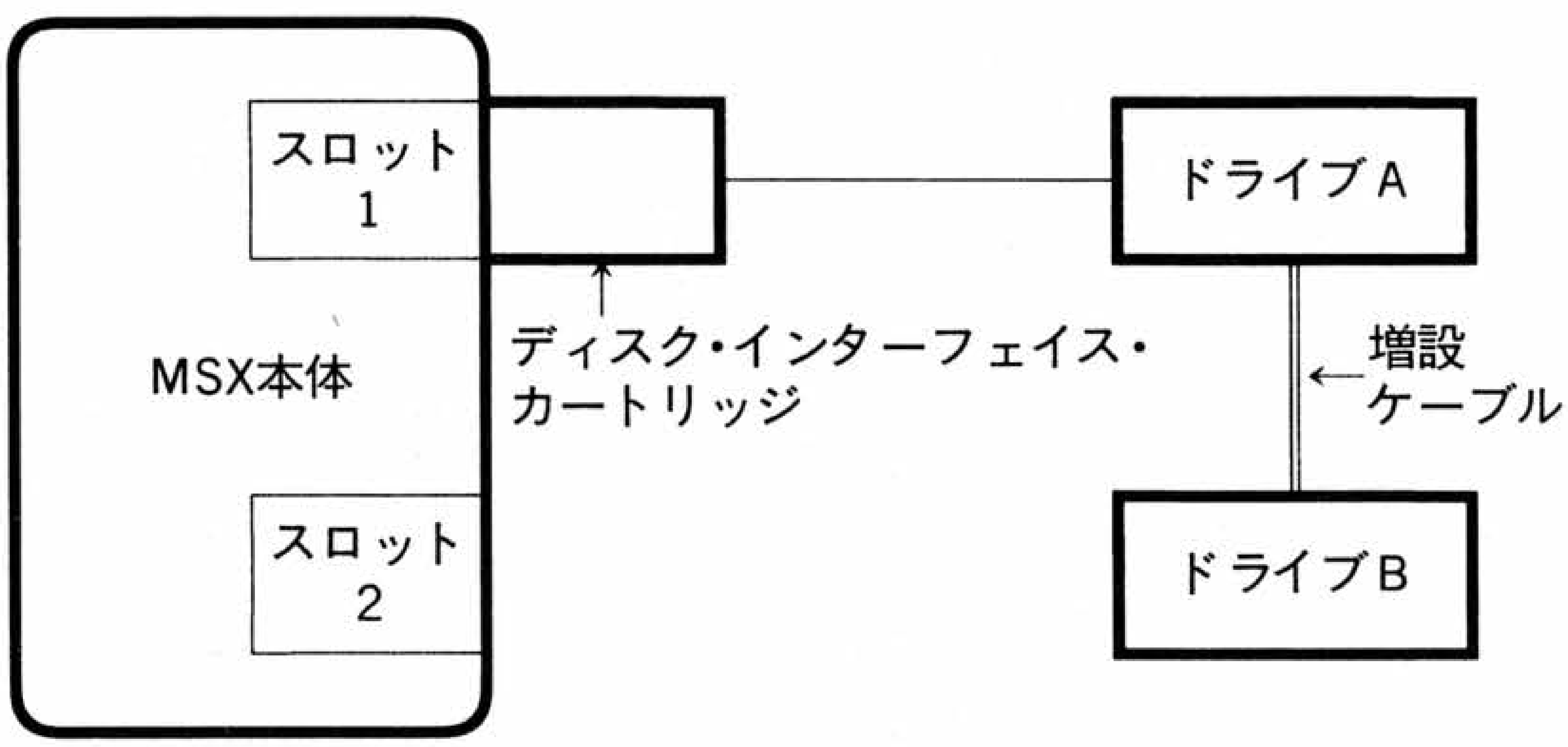
▶２台のディスク・インターフェイス・カートリッジに
それぞれ外付け型ドライブを１台ずつ接続した場合.



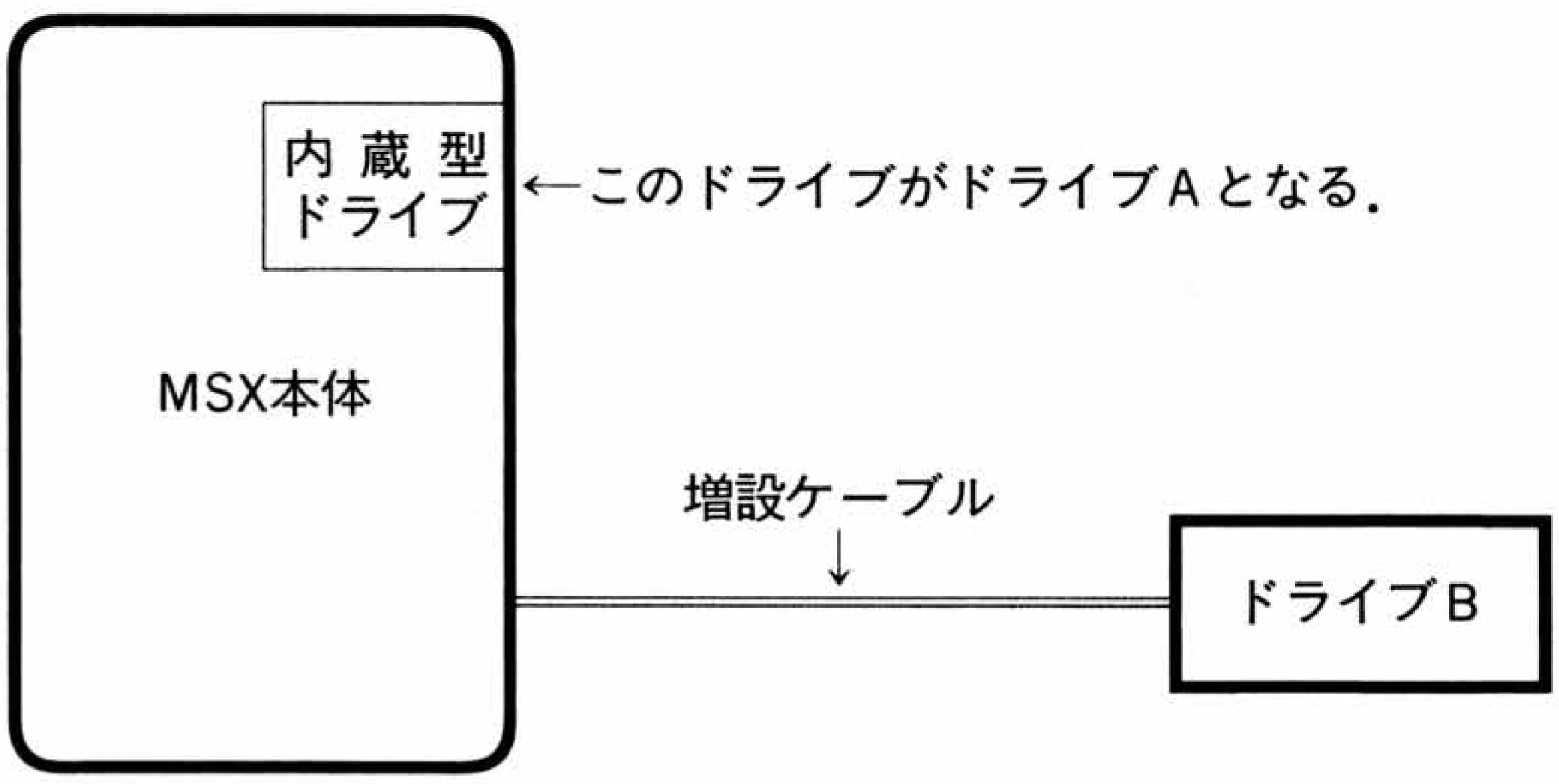
▶１台の内蔵型ドライブと，１台のディスク・インターフェイス・
カートリッジに外付け型ドライブを１台接続した場合.



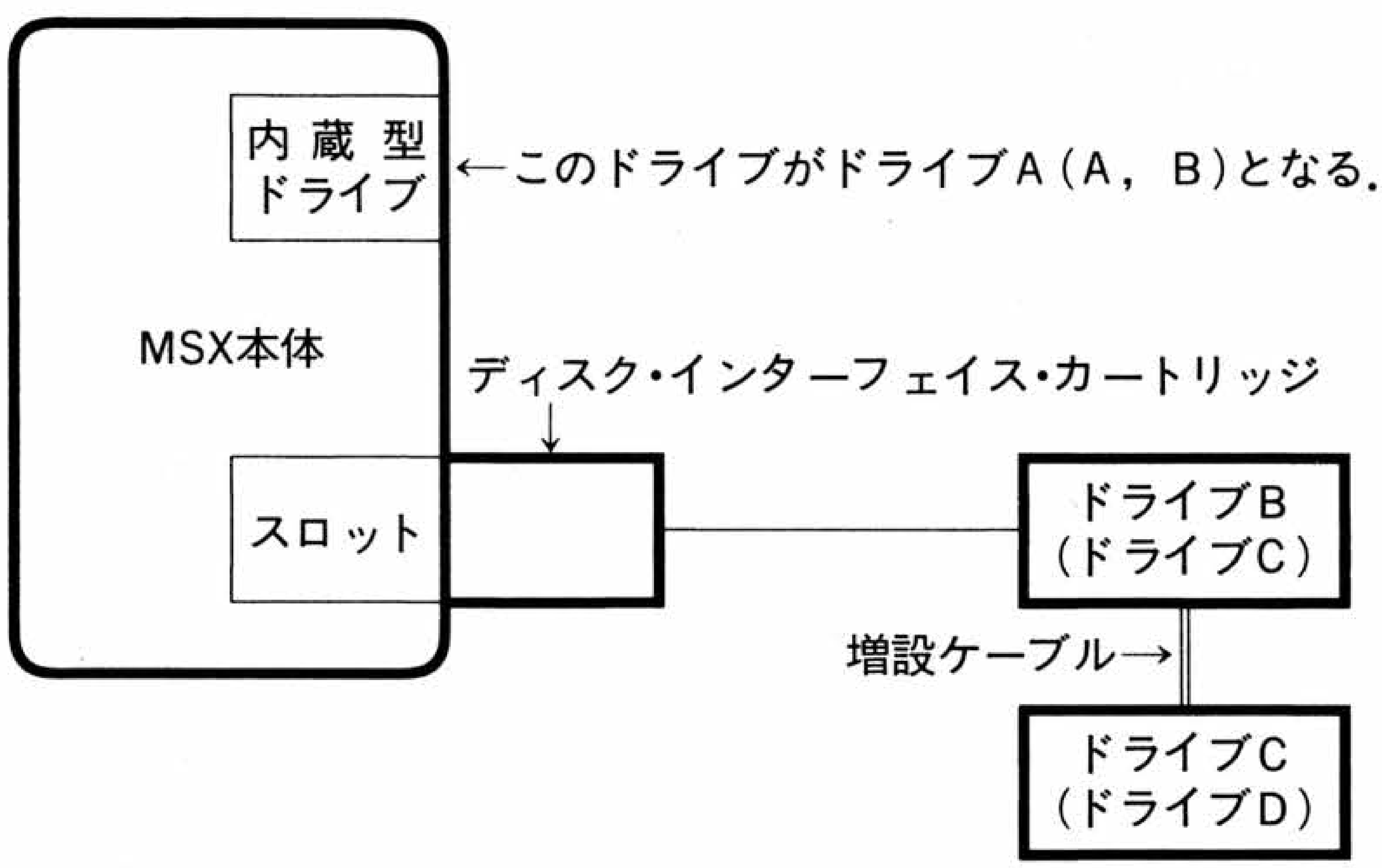
▶１台のディスク・インターフェイス・カートリッジに
外付け型ドライブを２台接続した場合.



▶ 1台の内蔵型ドライブと，増設ケーブルにより外付け型ドライブを1台接続した場合．



▶ 1台の内蔵型ドライブと，1台のディスク・インターフェイス・カートリッジに外付け型ドライブを2台接続した場合．



▶ 2台の内蔵型ドライブの場合．

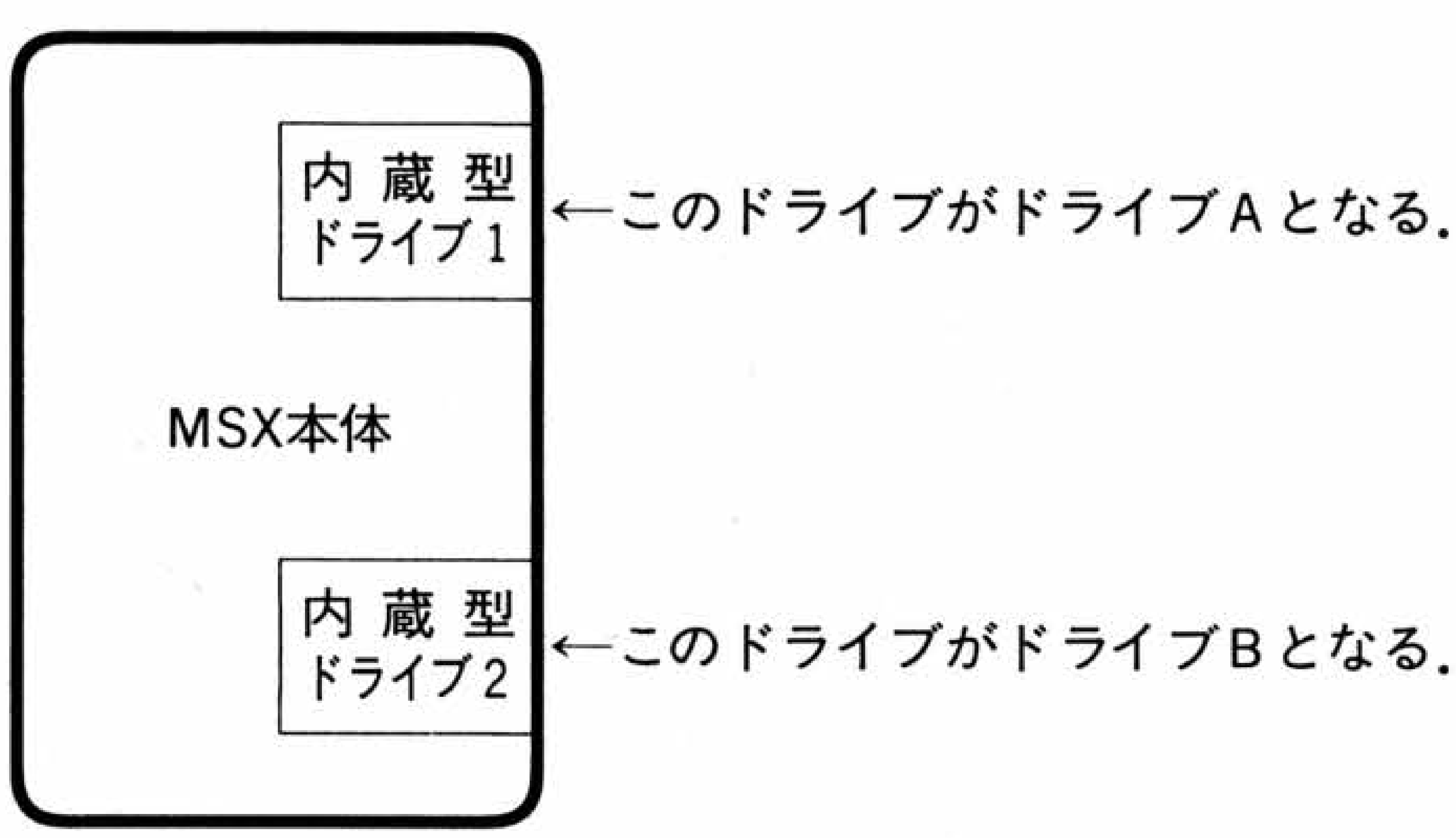


図2.1 ドライブのさまざまな組み合わせ

このように，ドライブを2台以上持っている場合，MSXとドライブの組み合わせによってはドライブBを使わずに，ドライブAとドライブCを使うといったケースも考えられます．

以下の説明ではドライブAとドライブBを使いますが，ドライブBを使わないシステム構成になっている人は，ドライブBをすべて自分のシステム構成で使用できるドライブとして読み換えてください．

それでは実際に起動するための確認を行いましょう．MSXとディスプレイ，ディスク・ドライブは正しく接続されていますか？　もし，きちんと接続されていないならば，MSXやディスク・ドライブの説明書を参照して正しく接続してください．

次に，MSX-DOSのシステム・ディスクを用意してください．ここでいうシステム・ディスクとは，ディスク・ドライブ・ユニットやディスク内蔵型MSXを購入時に付属してくるディスクのことを指します．

準備ができたら，まず周辺装置の電源を入れましょう．ここでは，システム・ディスクを使ってMSX-DOSを起動することにします．

では，システム・ディスクをドライブAに挿入してください．システム・ディスクをセットしたらMSXの電源を入れてください．これで，MSX-DOSがディスクから読み込まれます．MSX-DOSが起動すると，画面に図2.2のようなメッセージが表示されます．

```
MSX-DOS version 1.03
Copyright 1984 by Microsoft

COMMAND version 1.11
```

図2.2　DOSの起動メッセージ

MSXには，バッテリー・バックアップの付いた内蔵時計を持っている機種と持っていない機種があります．時計機能を持っていないMSXの場合，図2.3のようなメッセージが出て，日付を聞いてきますので入力してください．

入力の際の書式は，年(西暦)，月，日をそれぞれ〝−〟(マイナス記号)で区切ります．入力を間違えた場合には，[BS]がBASICと同様に機能しますから，リターン・キーを押すまえに訂正してください．

時計機能を持ったMSXの場合は，MSX-DOSが内蔵時計を自動的に参照するので，図2.3のようなメッセージは表示されません．

```
COMMAND version 1.11
Current date is Sun 84-01-01
Enter new date:1988-1-1
```

年，月，日をマイナス記号で区切る(年は下2桁だけでもよい)．

図2.3　日付の入力

さて，起動メッセージの表示および日付の入力がすむと，つづいて画面に，**図2.4**のような記号が表示されます．

```
A>
```

図2.4　プロンプト

ここで表示されている〝A>〟という記号は**プロンプト**と呼ばれるものです．われわれ，ユーザーがDOSに対して与える命令をコマンドといいますが，プロンプトが表示され，すぐ右にカーソルが出ている状態は，MSX-DOSがコマンドを受け付けられる状態にあることを表しています．以下本書では，この状態を**コマンド・レベル**と呼ぶことにします．

あとで詳しく説明しますが，このプロンプトのAという文字は現在ドライブAを使っているということを示しています．

さあ，これであなたもMSX-DOSに対してコマンドを与えることができるのです．

2-2 バックアップ・ディスクを作成する

さて，MSX-DOS を起動させることはできましたが，MSX-DOS を使う限りシステム・ディスクは欠くことができません．この大切なシステム・ディスクがなんらかの事故で失われると，以後 MSX-DOS を起動させることができなくなるので，どこかに予備を作成しておく必要があります．この予備のディスクのことを**バックアップ・ディスク**と呼びます．本節では，ここでシステム・ディスクのバックアップ・ディスクを作成してみることにします．

ディスクについての基礎知識

具体的な作業にはいるまえに，これから使っていくフロッピー・ディスクについて，簡単に説明しておきましょう．MSX-DOS で使う **3.5 インチ・フロッピー・ディスク**は，**図 2.5** に示すように外見はプラスチックの四角い板ですが，中には磁性体を塗布されたポリエステルの薄い円盤がはいっています．この円盤が高速で回転しているところに，シャッターをスライドさせ，四角くあいている窓の部分から，磁気ヘッドが情報の読み書きを行います．

3.5 インチ・フロッピー・ディスクには，1D，2D，1DD，2DD といった種類がありますが，MSX では一般的に 1DD か 2DD を使用します．〝DD〞というのは，〝Double density, Double track（倍密度，倍トラック）〞の略であり，記録密度を示しています（p.30コラム参照）．また，1D や 2D を使用できる MSX もありますが，他の MSX との互換性を考えると使わない方がよいでしょう．

1DD と 2DD の違いはこの円盤の単面を使うか，それとも両面を使うかという点です．ですから，両面を使う 2DD の方がより多くの情報を記憶できることになります．

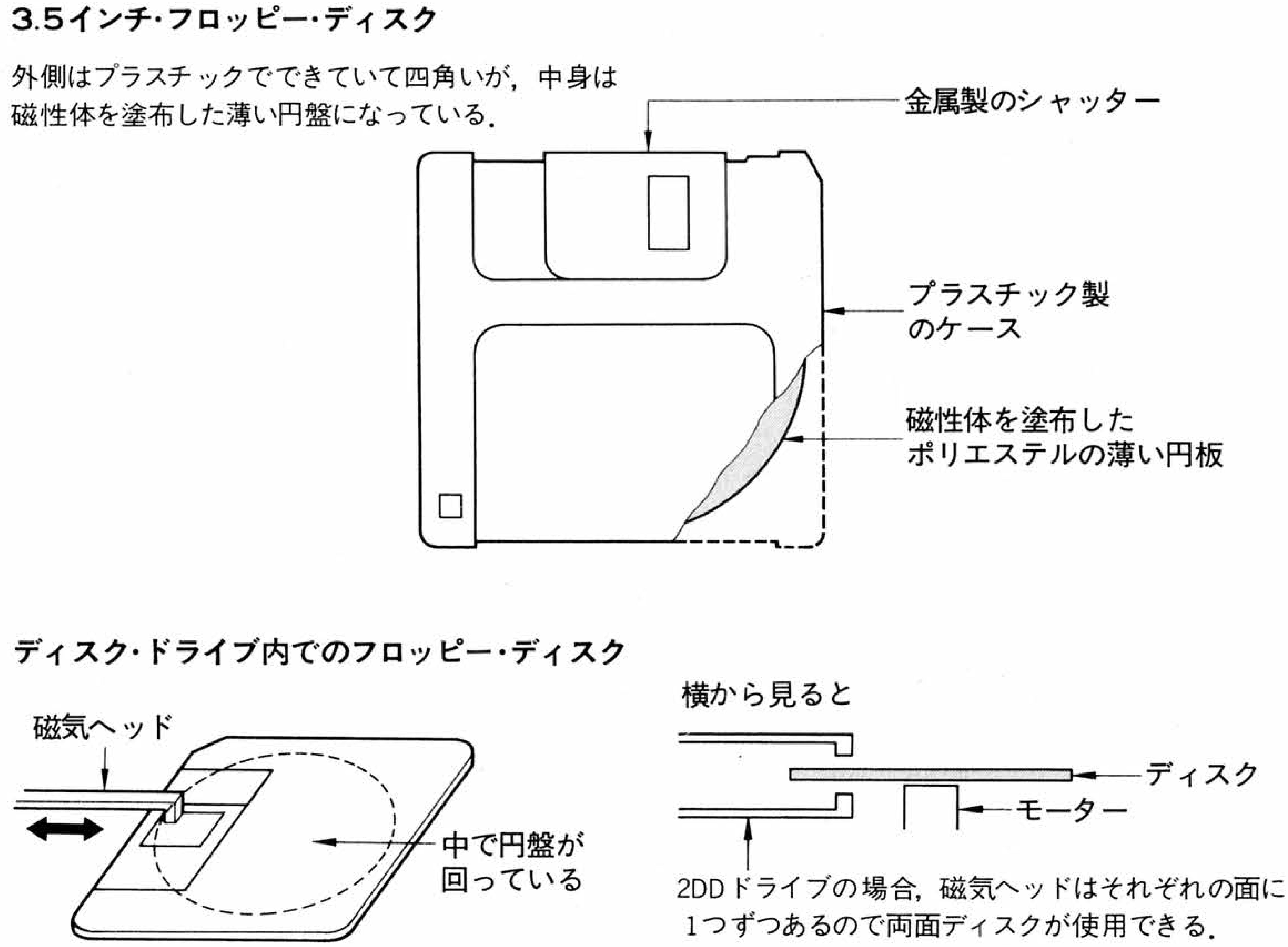


図2.5　3.5インチ・フロッピー・ディスク

ディスクをフォーマットする

では，バックアップ・ディスク用に，新しいディスクを1枚用意してください．新しいディスクを使い始めるまえには，かならず**フォーマット**という作業をしなければなりません．フォーマットの意味は，p.30のコラムを参照してください．ここでは実際の作業について説明していきます．すでにデータを記録しているディスクをフォーマットすると，それ以前に記憶していたすべての情報が失われてしまいますから，間違ってもシステム・ディスクやアプリケーション・ディスクをフォーマットしないでください．

さて，フォーマットをするディスクの裏面右下を見てください．**図2.6**のように，小さなノッチが付いています．

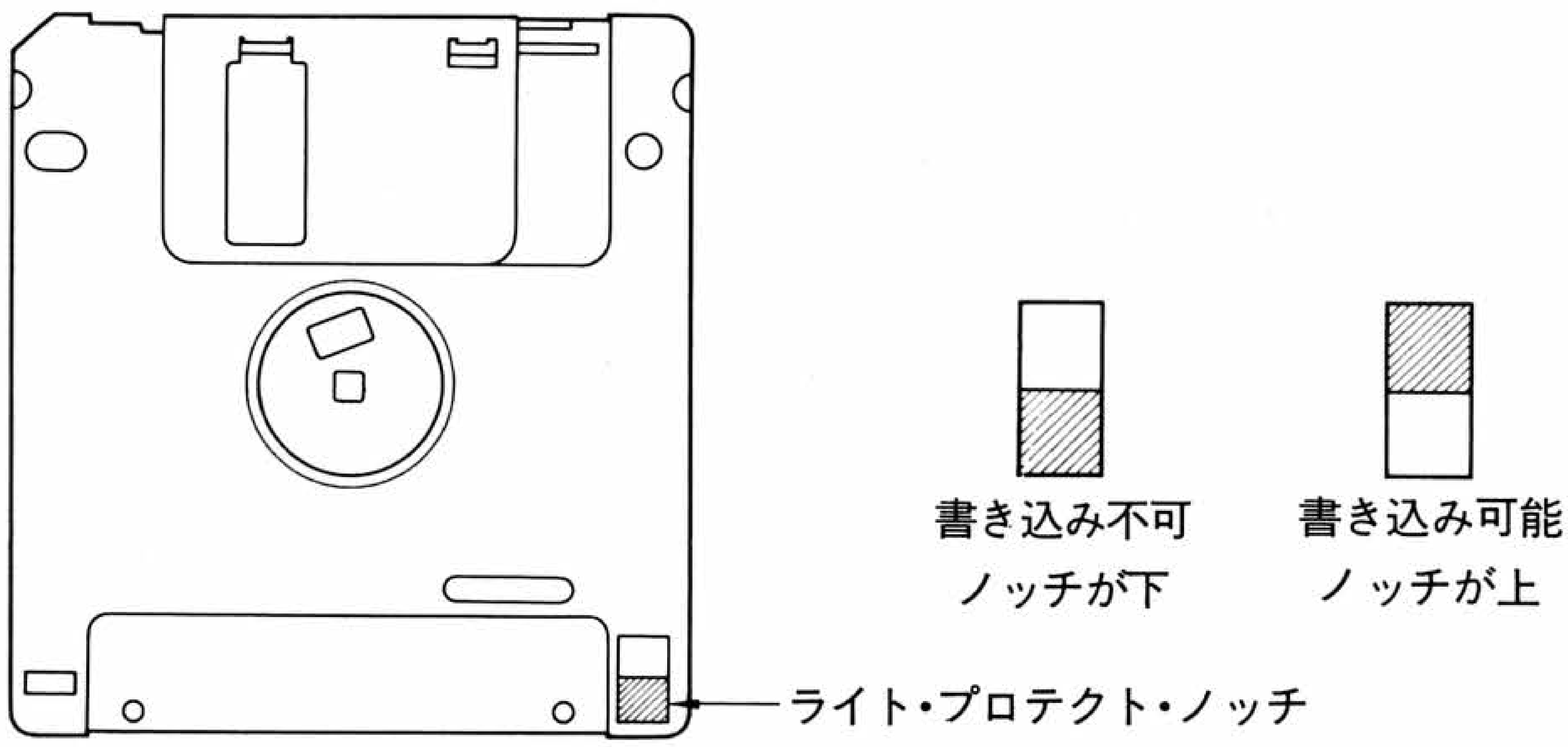


図 2.6　ライト・プロテクト・ノッチ

これは，**ライト・プロテクト・ノッチ**(Write Protect Notch)と呼ばれるものでノッチが下になっていると，ディスクへのデータの書き込みが禁止されます．カセット・テープでも爪を折ることによって録音を禁止できます．あれと同じで大切なデータを失わないように，ディスク上のデータの削除，上書きを禁止することができます．

ここでは，フォーマットをするので書き込めなくては困ります．ノッチが書き込み可能な状態になっていることを確認してください．

では，ドライブAからシステム・ディスクを取り出して，フォーマットするディスクを代わりに挿入してください．

MSX-DOS では，**FORMAT コマンド**によりフォーマットを行います．

ディスクをセットしたらコマンド・レベルから〝FORMAT〟と入力してください．DISK-BASIC を使っていた人ならば，このあとは，まったく同じ要領でフォーマットを行うことができます．

```
A>format⏎ ………………………… キーボードから入力する.
Drive name? (A,B) a ………… ドライブAのディスクをフォーマットする.
```

図 2.7　FORMAT コマンドの実行

FORMAT コマンドを実行すると，図 2.7のようなメッセージが表示され

て，どのドライブにはいっているディスクをフォーマットするのかを聞いてきます．ここでは，ドライブＡにはいっているディスクをフォーマットするので〝A〞と入力します．

次にフォーマットするディスクの種類を聞いてきますが，このメッセージはMSXによって違うので，**図2.8**とは少し異なるかもしれません．しかし，どのMSXにも1DDと2DDのフォーマットはあるはずです．あなたの用意したディスクが単面(1DD)か両面(2DD)かに合わせて，画面に表示される番号を正しく選択し，入力してください．

```
1 - 1 side
2 - 2 sides
3 - 1 side, double track
4 - 2 sides,double track

? 4
```

（1〜4の表示）この表示はMSXによって異なる．

（? 4）……………この例では，2DDのディスクをフォーマットしている．

図2.8　ディスクの種類の選択

ディスクの種類の選択が終わると，画面に〝Strike a key when ready〞というメッセージが出力されます．これは「準備ができたら，何かキーを押してください」という意味で，これからたびたび登場しますから，覚えておいてください．では，適当なキーを押してください．これでフォーマットが始まります．MSXはしばらくディスクをアクセスし続け，コマンドを受け付けなくなります．

しばらくすると**図2.9**のようなメッセージが出力され，フォーマットは終了します．

```
Strike a key when ready
Format complete

A>
```

（Strike a key when ready）………このメッセージが表示されたら，任意のキーを押すことにより，フォーマットが開始される．

（Format complete）……………フォーマット終了のメッセージ

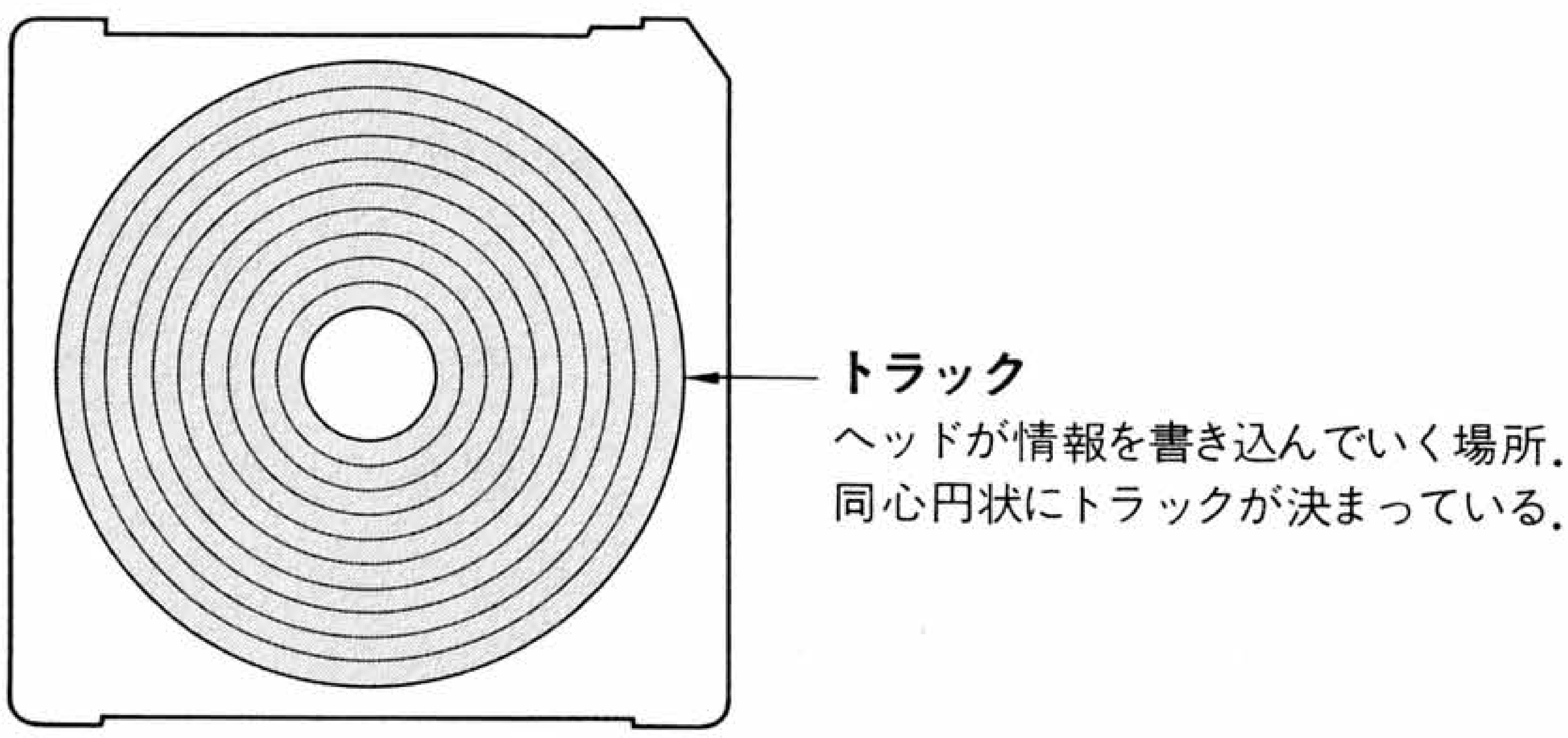
図2.9　フォーマットの終了

フォーマットが終了すると再び，コマンド・レベルに戻り命令を受け付けるようになります．

FORMAT コマンドは，新しいディスクを使うまえにかならず行わなくて

コラム1　MSX-DOSのディスク・フォーマット

ディスクの上のデータは，ディスクの中の磁性体円盤に同心円状に書き込まれていくのですが，円盤のどこにでも記録されるというのではなく，特定の円周に沿って記録されます．この円周を**トラック**と呼びます．運動場でランナーが走る道をトラックと呼びますが，これと同じように磁気ヘッドが走る道もトラックと呼ぶのです．ディスクに情報を記録するにはトラックが必要なのですが，買ったばかりのディスクにはまだ，トラックが決まっていません．ちょうど整備していない運動場のようなものです．



そこで，まだまっさらなディスクにトラックを引いてやらなくてはなりません．この作業のことをフォーマットと呼びます(フォーマットによって決まる記憶形式のこともフォーマットと呼びます)．

このフォーマットはDOSによって異なるものですから，たとえば，CP/MとMS-DOSでは1枚のディスクを互いに読み書きすることはできません．ところが，MSX-DOSとMS-DOSでは同じフォーマットを使っているために1枚のディスクを互いに読み書きすることが可能になっています．

フォーマットは新しいディスクを使用するまえに1回だけ行えば，それ以降は同じDOSを使っている限り再びフォーマットする必要はありません．

はならない作業なので，これから何回も使うことになります．きちんと使えるようにしておいてください．

MSX-DOSのフォーマットには，次の4種類があります．

	1DD 9セクタ	2DD 9セクタ	1DD 8セクタ	2DD 8セクタ
記録可能な総ファイル数	112	112	112	112
FATが占めるセクタ数	2	3	1	2
1トラックのセクタ数	9	9	8	8
使用可能な面	単面	両面	単面	両面
片面あたりのトラック数	80	80	80	80
1セクタのバイト数	512	512	512	512
使用可能なセクタ数	708	1426	630	1268
使用可能な総バイト数	362496	730112	322560	649216

1DDと2DDの違いはまえにも述べたように，単面を使うか両面を使うかの違いです．これは，用意したディスクに合わせてフォーマットすればよいのですが，8セクタとか9セクタとかいう見なれない言葉がでてきました．

セクタとはトラックをさらに細かく分けた単位のことです．ディスクをトラックに分けて管理するのはよいのですが，これではディスクをきめ細かく管理できません．MSX-DOSでは，片面を80トラックに分けていますが，さらに細かく管理するために1トラックを8セクタもしくは，9セクタに分けて管理することにしています．

8セクタ・フォーマットは使用できないMSXもあり，使用できるディスクの容量も減ってしまうので，普通は9セクタ・フォーマットを使います．

このため，MSX-DOSで主に使用されるのは，1DDか2DDの9セクタ・フォーマットということになります．

MSX-DOSは1セクタに512バイト記憶するので，1DDならば360Kバイト(512×9×80バイト)，2DDならばその倍の720Kバイトの記憶容量を持つことになります．

システム・ディスクを作る

さて，いよいよシステム・ディスクのバックアップ・ディスクを作成することにしましょう．具体的には，システム・ディスクの記録内容を，MSX-DOSのコマンドを使って，用意した新しいディスクにコピーします．ここでは，ディスク・ドライブを2台以上持っている場合と1台だけの場合に分けて説明します．

●ディスク・ドライブを2台以上持っている場合

まず，システム・ディスクのライト・プロテクト・ノッチを書き込み禁止にしてください．これは，誤ってもとのディスクの内容を消去してしまうことを避けるためです．

次にシステム・ディスクをドライブAに，さきほどフォーマットしたディスクをドライブBに挿入してください(図2.10)．

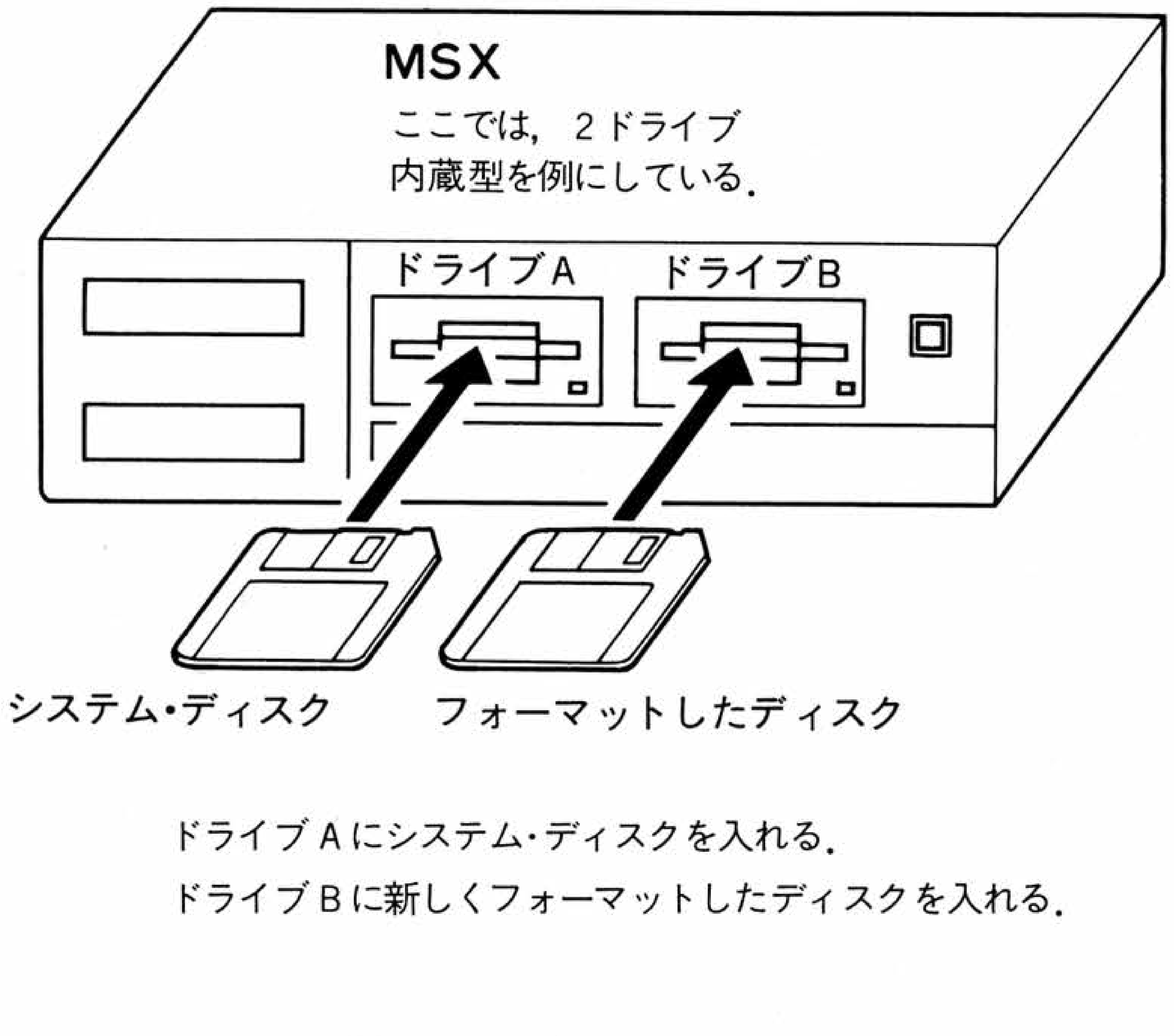


図2.10　ディスクのセット

次にコマンド・レベルから，**図 2.11** のように入力してください．リターンを押すまえならば [BS] が BASIC と同様に機能するので，訂正することができます．さきほども述べましたが，2 台目以降のドライブが B でない人は，自分のシステム構成に応じたドライブに対してコピーしてください．

```
A>copy a:msxdos.sys b:⏎
                    ↑
                    └──2台目のドライブがB以外の場合は，ここを書き変える．
```

図 2.11　MSXDOS.SYS のコピー

図 2.11 で実行したコマンドの意味については 3 章で詳しく解説しますが，ここで簡単に説明しておくと，「ドライブ A にはいっているディスクの，MSXDOS.SYS というファイルを，ドライブ B にはいっているディスクにコピーしなさい」ということになります．

MSXDOS.SYS のコピーに成功すると**図 2.12** のようなメッセージが表示されます．

```
        1 file copied ……………コピー終了のメッセージ
A>
```

図 2.12　コピー終了

同様にして，COMANND.COM のコピーを行います(**図 2.13**)．

```
A>copy a:command.com b:⏎
        1 file copied
A>
```

図 2.13　COMMAND.COM のコピー

●ドライブが1台の場合

この場合はMSX-DOSの**仮想ドライブ機能**を使ってファイルをコピーします．仮想ドライブ機能とは，**図2.14**に示すように，1台のディスク・ドライブをドライブA，ドライブBで切り替えて使う機能のことです．

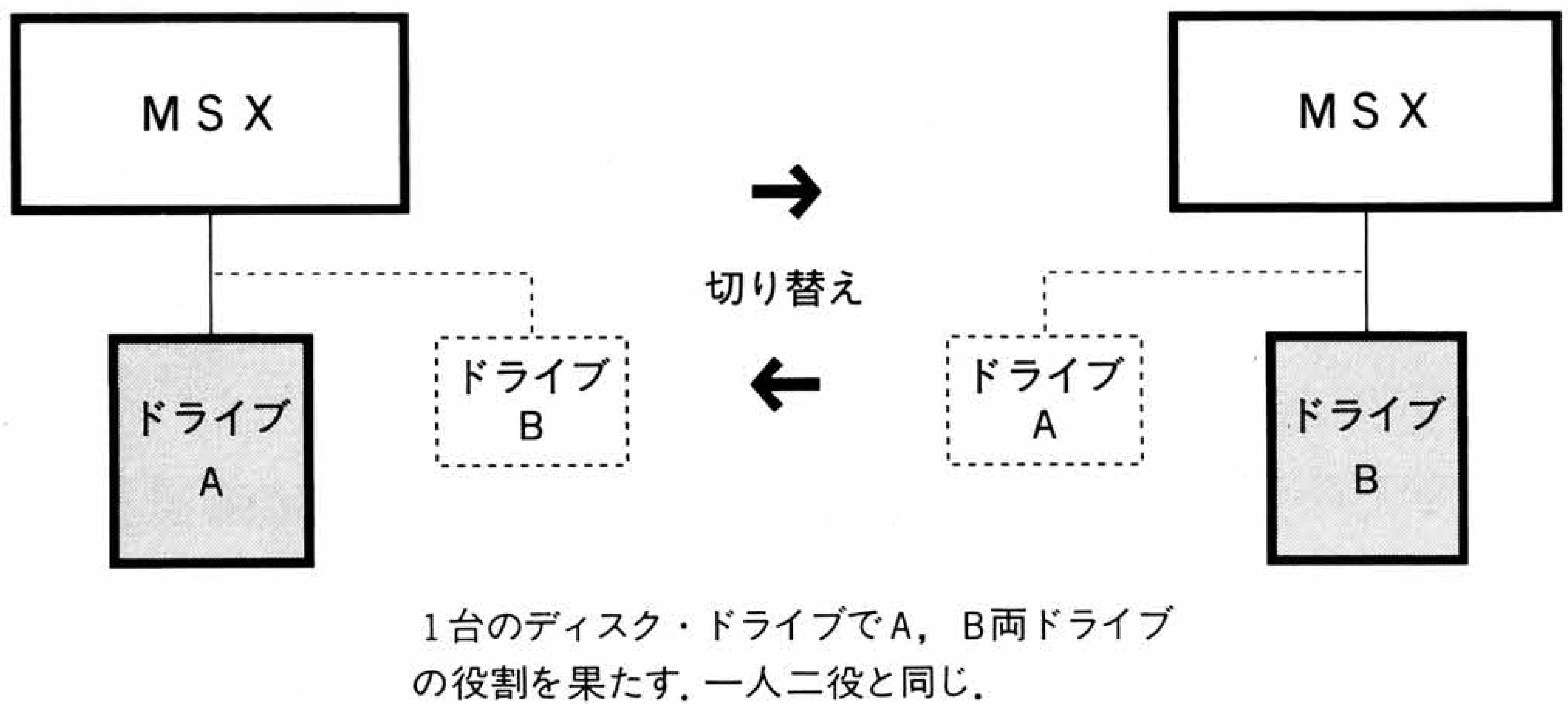


図2.14　仮想ドライブ機能

MSX-DOSでは，ディスク・インターフェイス・カートリッジにドライブが1台だけ接続されている場合には，標準で仮想ドライブ機能が動作します．

まず，**ソース・ディスク**(コピー元：ここではシステム・ディスクを指します)のライト・プロテクト・ノッチを書き込み禁止にしてください．仮想ドライブ機能を使う場合，とくに慣れないうちは，ディスクの差し替えを誤ってソース・ディスクの記録内容を書き変えてしまう恐れがあります．そのようなことを避けるためにも，ソース・ディスクのライト・プロテクト・ノッチは書き込み禁止にすることを習慣とするべきです．

まず，ソース・ディスクであるシステム・ディスクをディスク・ドライブに挿入してください．与える命令自体は2ドライブのときと，まったく同じです．しかし，仮想ドライブ機能のもとでは，ディスク・ドライブからしばらくデータを読み込んだのちに，**図2.15**のようなメッセージが表示されます．

```
A>copy msxdos.sys b: ⏎

Insert diskette for drive B:    このメッセージが表示されたら，さき
and strike a key when ready     ほどフォーマットしたディスクと差し
                                替えて任意のキーを押す．
      1 file copied ………………コピーが終了すると，このメッセージ
A>                              が表示される．
```

図 2.15　1 ドライブの場合のディスク交換メッセージ

このときすでに，ディスク・ドライブはドライブ B に切り替わっているので，このメッセージに従って，ソース・ディスクを取り出して，ドライブ B に入れるべき，さきほどフォーマットした**デステイネーション・ディスク**(コピー先：ここではさきほどフォーマットしたディスク)を代わりに挿入し，何か適当なキーを押します．すると，ドライブ B に対して，ドライブ A から読み込んでおいたデータを書き込みます

最終的には，コピーが終了すると，ディスク・ドライブはドライブ A に切り替わり，コマンド・レベルに戻ります．

図 2.14, 図 2.15 を参考にして，COMMAND.COM もコピーしてください．

さて，こうして MSXDOS.SYS と COMMAND.COM がコピーできればシステム・ディスクのバックアップ・ディスクが作成できたことになります．

ドライブ A に新しく作ったシステム・ディスクを入れてリセット・ボタンを押してください(リセット・ボタンがない場合は電源を 1 度切る)．正常に起動すれば，バックアップ・ディスクの作成は成功です．これ以降では，このバックアップ・ディスクを使いますので，もとのシステム・ディスクは大切に保管しておいてください．

もし，バックアップ・ディスクの作成に失敗していたら最初からやり直してください．落ち着いてやればかならずうまくいくはずです．

コラム2　MSX-DOSが起動する仕組み

いままでは，ディスク・ドライブに付属しているMSX-DOSを起動させるディスクのことをシステム・ディスクと呼んできましたが，DOSを起動させることのできるディスクはすべて，広い意味でシステム・ディスクと呼びます．

たとえば，DOSを起動させ，実行されるワープロなどのアプリケーション・ディスクがあれば，それもシステム・ディスクの一種であるということができます．

さらに詳しく述べるならば，システム・ディスクとは，DOSシステムと呼ばれるDOSの本体に当たるプログラムを記録しているディスクのことです．このDOSシステムが記録されていないと，DOSを起動することのできるディスクにはなりません．これは，DOSもプログラムなので，起動されるまえはディスク上に記録されおり，そこから読み込まれて起動されるためです．

一方，システムをディスク上に持っていないディスクは，それ単体ではDOSを起動させることはできません．つまり，システム・ディスクを使ってDOSを起動させたあとでないと使用できないのですが，このようなディスクをデータ・ディスクと呼びます．

データ・ディスクは，システム・ディスクのようにDOSを起動することができないので不便といえば不便ですが，システムを記録する必要がない分，より多くのデータを記録することができます．

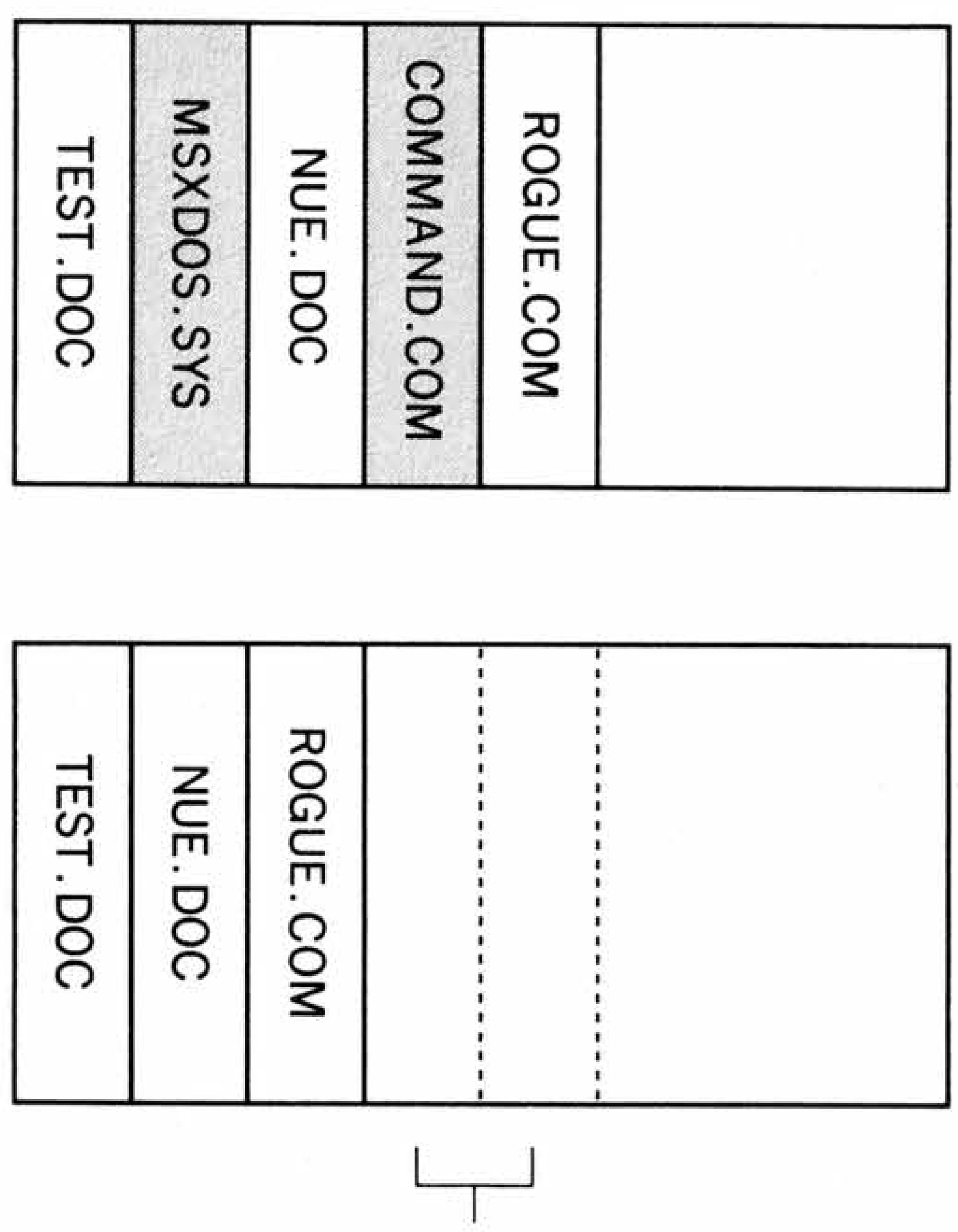
MSX-DOSでは，MSXDOS.SYSとCOMMAND.COMというファイルがシステムに当たります．これらの2つのファイルを記録さえすれば，いつでもデータ・ディスクをシステム・ディスクにすることができます．



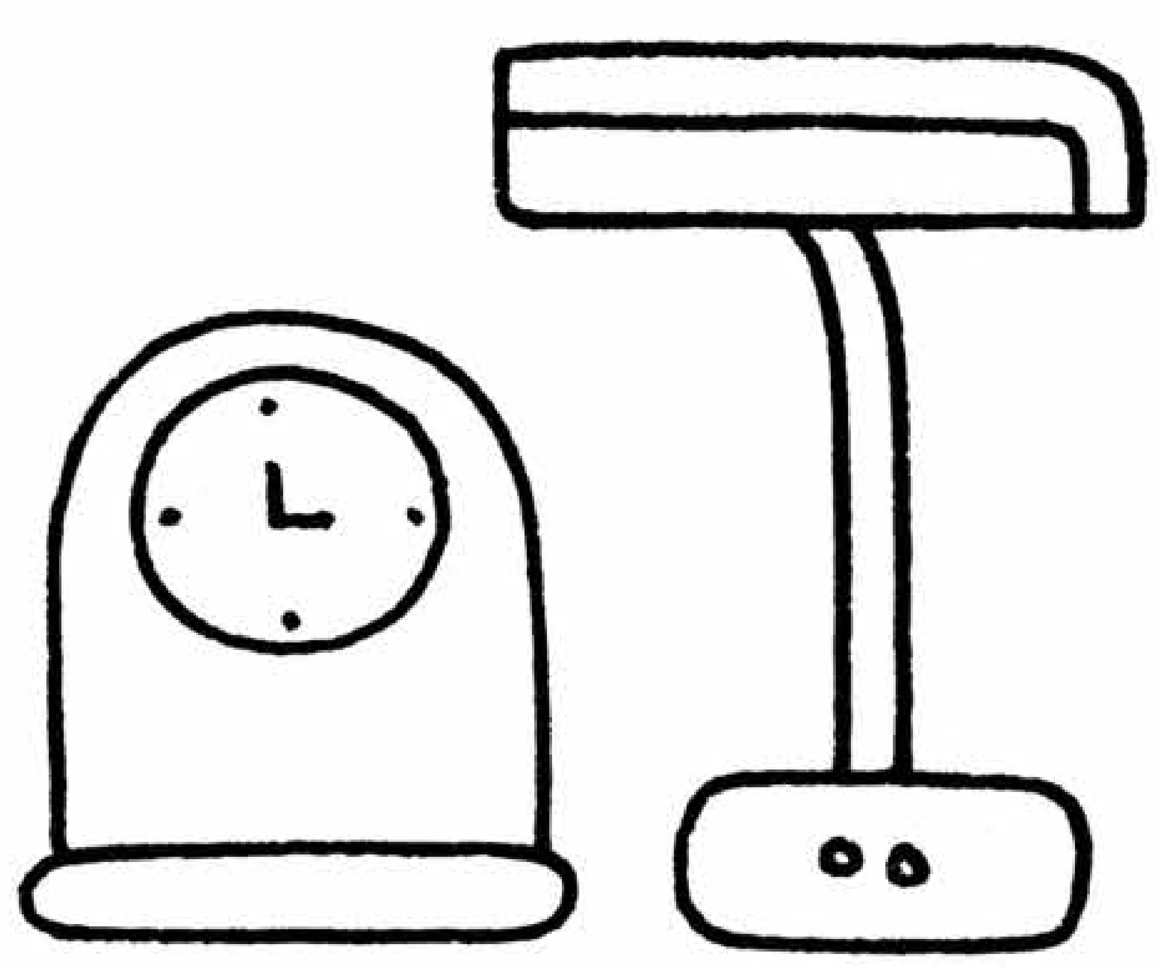
ディスクの中に　MSXDOS.SYS
COMMAND.COM　があると，MSX-DOSが起動する.

システム・ディスクの内容

データ・ディスクの内容



あとからいつでもMSXDOS.SYSとCOMMAND.COM
を書き込むことができる.



2-3 コマンドラインの編集・訂正

本節では，MSX-DOS でのコマンドの入力方法について説明していきましょう．MSX-DOS 上でのコマンドの入力方法は，BASICとはかなり違います．

まず，MSX-DOS では，BASIC のように，画面のあちこちにカーソルを移動させて，入力，訂正できるスクリーン・エディット機能というものはありません．そのためコマンドは，現在入力中の1行の間だけでしか訂正することはできません．

最初は少し不便に感じるかもしれません．しかし，本節で説明する**テンプレート**と呼ばれる機能を併せて使えば，非常に使いやすく思えてくるはずです．

テンプレート

プロンプト〝A>〟からあとの，コマンドを入力できる行を**コマンド・ライン**といいます．そして，テンプレートとは，キーボードからコマンド・ラインに入力した文字をいったん記憶しておく場所のことです．つまり，**図 2.16** に示すように，直前に入力されたコマンドを続けて実行したり，似たようなコ

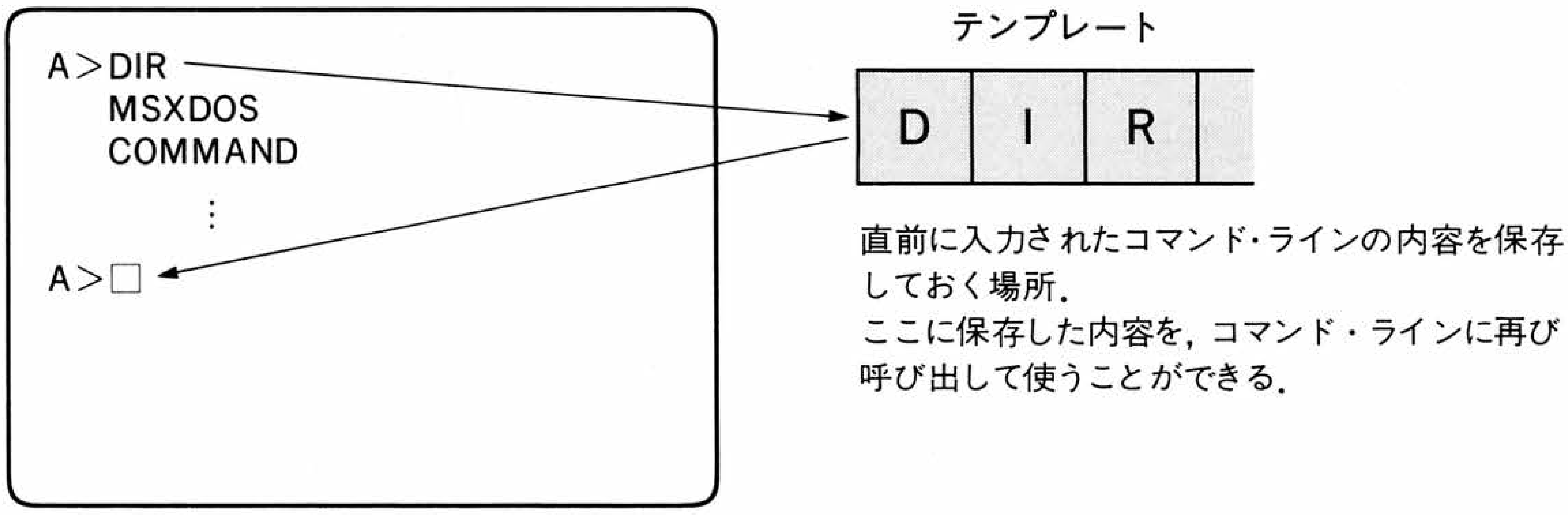


図 2.16　テンプレート概念図

マンドを少し修正して実行したりする場合には，テンプレートから流用可能な部分を呼び出してきて使うことにより，全部を打ち直す必要はなくなります．

では，テンプレートを実際に使ってみることにしましょう．MSX-DOSには，〝DIR〃というコマンドがあります．このコマンドの詳しい意味はあとで説明しますが，ここではディスクの記録内容を確認するコマンドだということだけを覚えておいてください．

さて，ここでは〝DIR　*.COM〃を実行するつもりが，キー操作を間違えて〝DRI　*.COM〃と入力してしまったことにします．では，**図2.17**のようにコマンド・ラインから〝DRI　*.COM〃と入力してください．

```
A>dri *.com ⏎
Bad command or file name ……………エラー・メッセージが表示される.
A>
```

図2.17　間違ったコマンドを入力

MSX-DOSにはDRIというコマンドはないので，図2.17のようにエラーメッセージを表示しています．ここで，テンプレートを使って〝DIR　*.COM〃に直してみましょう．作業の手順を，**図2.18**に示します．

まず，[CTRL]+[¥]または[→]を押して，テンプレートからコマンドラインに1文字だけコピーする．

```
A>d                    現在のテンプレート    d r i  * . c o m
                                             ↓[CTRL]+[¥]または[→]を押す.
                       コマンド・ライン      d
```

↓

次に，間違っている〝ri〃を〝ir〃に直すため，キーボードから[I]，[R]と続けて入力する．

```
A>dir                  現在のテンプレート      r i ┆  * . c o m
                       コマンド・ライン      d i r ┆
                                               ┆……キーボードから入力
```

↓

残りの"␣＊.com"はそのままテンプレートからコピーすればよいので，[↓]を押す．

```
A>dir *.com          現在のテンプレート          ＊.com
                                                  ↓[↓]を押す．
                     コマンド・ライン      dir    ＊.com
```

図2.18　テンプレートによる修正1

正しくコマンドを訂正できたので，リターンキーを押してDIRコマンドを実行させます．**図2.19**のようになります．

```
A>dir *.com ⏎
COMMAND  COM    6656 85-09-02 10:10p  ……正しく修正されていれば，このように表示される．
       1 file   719872 bytes free
A>
```

図2.19　修正したコマンドを実行

テンプレートには，まだこれ以外の機能もあります．

次の例として，"DIR　MSXDOS.SYS"と入力したかったのに，"DIR MSDOS!.SYS"とキーをタイプしてしまったところで間違いに気付いたことにします．では，これを**図2.20**のようにテンプレート機能を使って修正してみましょう．

現在コマンド・ラインにある文字を，テンプレートにコピーするには[HOME]を使う

```
A>dir msdos!.sys@            テンプレート     dir msdos!.sys
                                                  ↑[HOME]を押す．
  └─[HOME]を押す．           コマンドライン   dir msdos! sys
```

↓

[HOME] を押すと，コマンドラインにある文字がテンプレートにコピーされたしるしとして，@が表示され，改行される．
[SELECT] を押した直後に文字を入力すると，テンプレートの先頭から入力された文字の直前までをコマンドラインにコピーするので，これを利用する．

```
A>dir msdos!.sys@        テンプレート      dir msdos!.sys
  dir ms                                   ↓ [SELECT] を押した後 [D] を押す.
                         コマンドライン    dir ms
```

↓

テンプレートの内容を壊さずに，文字を挿入するには，[INS] を使う．

```
A>dir msdos!.sys@        入力前のテンプレート      dos!.sys
  dir msx
                         入力後のテンプレート       dos!.sys
  [INS] を押した直後にキ
  ーから入力した文字数    コマンドライン      dir msx ←┐
  だけテンプレートの内容
  もずれていく.                   [INS] を押したあとキーボードから入力.
```

↓

“！”の直前まではそのまま使えるので，[SELECT] につづけて [!] を押す．

```
A>dir msdos!.sys@        テンプレート              dos!.sys
  dir msxdos                                       ↓ [SELECT] を押したあと
                         コマンドライン   dir msxdos          [!] を押す.
```

↓

“！”はコピーしたくないので，[DEL] を押して1文字スキップする．
残りの“.sys”はそのまま使えるので，[↓] でコピーする．

```
                                                   !.sys
A>dir msdos!.sys@                                    [DEL] を押す.
  dir msxdos.sys         テンプレート              .sys
                                                   ↓ [↓] を押す.
                         コマンドライン   dir msxdos.sys
```

図 2.20 テンプレートによる修正 2

またこれまででも使いましたが，コマンド・ラインで最後に入力した1文字を削除するには [BS] を使います．[BS] の代わりに [←] を使うこともできます．また，[↑] を使えば，1文字だけではなくコマンド・ラインの文字すべてを削除することができます(**図 2.21**)．

```
           [BS] または [←] を押す.
A>abcd ──────────→ A>abc
           [↑] を押す.
A>abcd ──────────→ A>
```

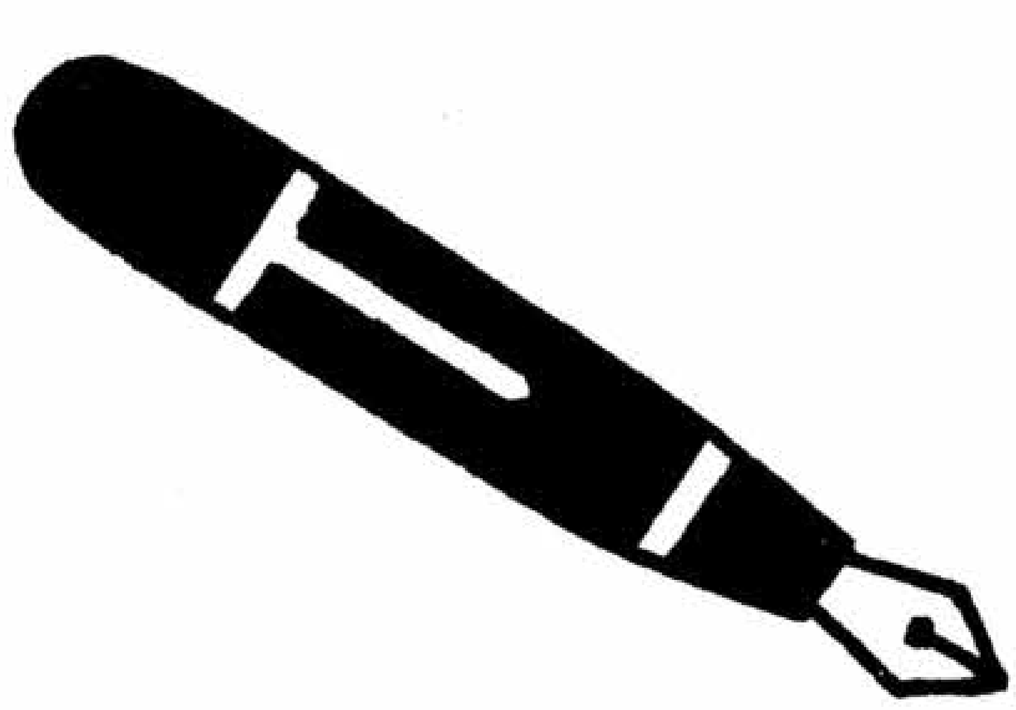
図 2.21　コマンド・ラインの文字の削除

またテンプレートとは関係ないのですが，[CTRL]＋[J]で画面上だけでの改行を行います．これは，単に画面上で改行されているだけで実際にリターンを押したことにはなっていないので気を付けてください．**図 2.22** に実行例を示します．

```
A>dir [CTRL]+[J]          [CTRL]+[J]を押すと改行されるが，コマンドラインはつながって
 *.com [⏎]                 いるので，“dir ＊.com”としてコマンドは実行されている．
COMMAND  COM    6656 85-09-02 10:10p
        1 file   719872 bytes free
A>
```

図 2.22　画面上だけでの改行

以上でコマンド・ラインとテンプレートに関するキー操作の説明は終わりです．いちおう見やすいように**表 2.1** にまとめておきましょう．1つの機能に複数のキーが割り当てられている場合には，どれを使っても同じです．



機　能	特殊キー	内　　　容
1字コピー	[→] [CTRL]＋[¥]	テンプレートからコマンド行へ1文字コピー
指定コピー	[SELECT] [CTRL]＋[X]	テンプレートからコマンド行へ，指定された文字の直前までの文字をコピー
1行コピー	[↓] [CTRL]＋[SHIFT]＋[_]	テンプレート内に残っているすべての文字をコマンド行へコピー
1字スキップ	[DEL]	テンプレート内で1文字飛び越し（コピーしない）
指定スキップ	[CLS] [CTRL]＋[L]	テンプレート内の文字を指定された文字の直前まで飛び越し（コピーしない）
[取消]	[↑] [ESC] [CTRL]＋[^] [CTRL]＋[[] [CTRL]＋[U]	現在のコマンド・ラインの取り消し（テンプレートの内容は変更されない）
[挿入]	[INS] [CTRL]＋[R]	挿入モードの切り換え
[後退]	[←] [BS] [CTRL]＋[H] [CTRL]＋[]]	コマンド行から最後の1文字を削除
新しい行を作る	[HOME] [CTRL]＋[K]	コマンド行をテンプレートにコピー（新しいテンプレートを作成）
行替え	[CTRL]＋[J]	ディスプレイの表示のみ改行

表 2.1　テンプレート関係

2-4 ディスクの内容を表示する

さてMSX-DOSのコマンドの入力方法も説明し終わりましたから，いよいよコマンドの説明にはいりましょう．一番最初は，さきほども使った**DIRコマンド**です．

DIRはDIRectoryの略で住所録とか電話帳などという意味があります．ディスクに記録されているファイルは，そのファイルの名前や作成された日時などの情報を**ディレクトリ**と呼ばれる部分に登録されています．つまりディレクトリとはファイルの名簿のようなもので，このディレクトリを表示する命令がDIRコマンドなのです．このDIRコマンドを実行することを〝ディレクトリを表示させる〞といいます．

では，**図2.23**のように，コマンド・ラインで〝DIR〞と入力してディレクトリを表示させてみてください．DIRのあとに何も付けないと，MSX-DOSが現在管理しているディスクに記録されている，すべてのファイルについてディレクトリが表示されます．

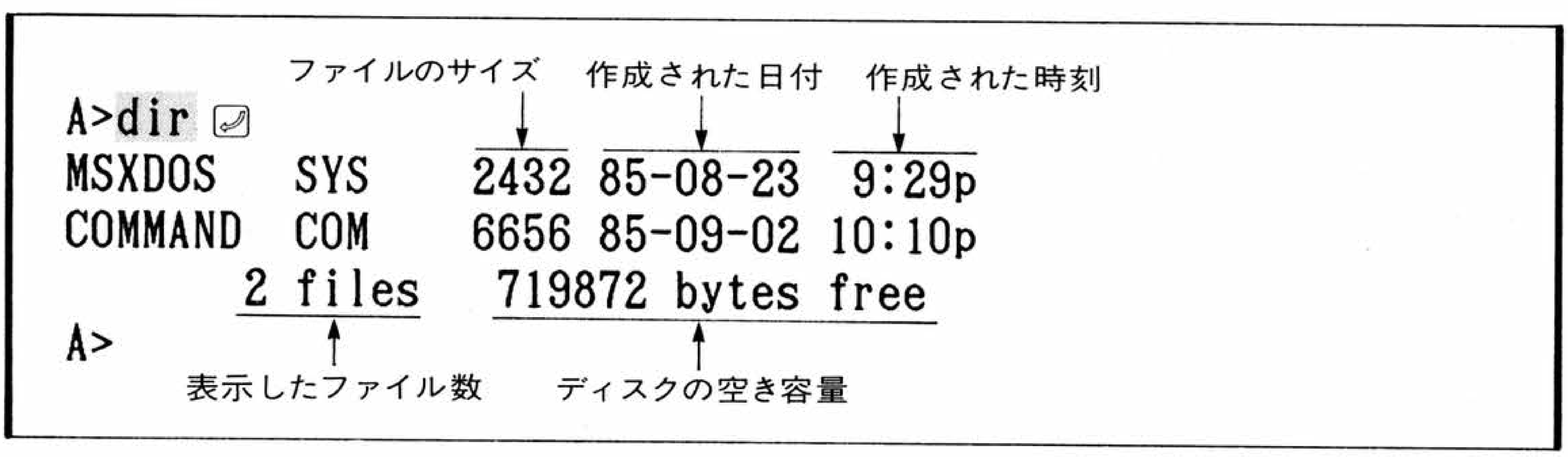


図2.23 ディレクトリの表示

DIRコマンドを実行すると，ファイル名，ファイルのサイズ(バイト単位)，ファイルの作られた日付，時刻が表示されます(ただし，画面幅を35文字以下に設定すると，時刻は表示されません．同様に28文字以下で日付，19文字

以下でファイルのサイズが表示されなくなります．画面幅の再設定は5章で解説します)．そして，最後に表示したファイル数と，ディスクの空き容量を併せて表示します．

また，**図2.24**のようにDIRのあとにファイルの名前を付けると，その名前のファイルを探し，もし見つかればそのファイルのディレクトリを表示します．

```
A>dir command.com ⏎
COMMAND  COM    6656 85-09-02 10:10p
       1 file   719872 bytes free
A>dir nue.doc ⏎
File not found
A>
```

……ファイルが見つかった場合はディレクトリを表示する．

……ファイルが見つからない場合はこのようなメッセージが表示される．

図2.24　ファイル名を指定したDIRコマンド

また，DIRコマンドはドライブAだけでなく，他のドライブに挿入されているディスクのディレクトリを表示させることも可能です．

ディスク・ドライブを2台以上持っている人は，ドライブBに何か記録されているディスクを入れてください(システムのオリジナルでも，何かのアプリケーション・ディスクでも，DISK-BASICのプログラムのはいったディスクでもかまいません)．

1台のディスク・ドライブで使っている人は，**図2.25**の命令を実行したあとで，仮想ドライブ機能のメッセージに従って何か適当なディスクを入れて

```
A>dir b: ⏎
MSXDOS   SYS     2432 85-08-23  9:29p
COMMAND  COM     6656 85-09-02 10:10p
AUTOEXEC BAT       77 87-02-12  3:58a
M80      COM    20480 85-07-11  6:46p
L80      COM    10752 85-07-11  7:01p
            :
```

この例では，ドライブBにMSX-DOS TOOLSのディスクを入れてある．

図2.25　ドライブBのディレクトリを表示

ください．これ以降の操作では，とくに何も言わなくても仮想ドライブ機能が発動したらそのメッセージに従ってディスクを入れ替えてください．

図2.25でDIRの後ろに付いている〝B:〟は「ドライブBに対して」という意味です．MSX-DOSのコマンドでドライブ名を指定する場合には，後ろにコロン(：)を付けます．

ここで，**カレント・ドライブ**というものについて説明しましょう．カレント・ドライブ(Current Drive)とは，「現在のドライブ」という意味で，現在MSX-DOSが直接管理しているドライブを表します．

では，図2.22の操作例ではドライブAについてDIRコマンドを実行しているのになぜ，〝DIR　A:〟としなかったのでしょうか？

もちろん，〝DIR　A:〟としてもドライブAのディレクトリは正しく表示されます．しかし，MSX-DOSではカレント・ドライブについてはドライブの指定を省略できることになっているので，この場合はとくに指定しなくてもよいのです．

カレント・ドライブについてはドライブ指定を省略できるというのは，DIRコマンドに限ったことではなく，あとで紹介する他のコマンドについても言えることですので，よく覚えておいてください．

MSX-DOSでは，起動した直後はカレント・ドライブはドライブAになっています．では，カレント・ドライブをドライブBに変えてみましょう．カレント・ドライブを変更するには，**図2.26**のように変更先のドライブ名を入力します．

```
A>b: ⏎
B> ……………カレント・ドライブがドライブBになった.
```

図2.26　カレント・ドライブをBに変更

プロンプトを見てください．〝B>〟になっていますね．実は，プロンプトのアルファベットはカレント・ドライブを示していたのです．

今度は，この状態で後ろに何も付けずにDIRコマンドを実行してみましょう(**図2.27**)．

```
B>dir ⏎ ……………カレント・ドライブを省略.
MSXDOS   SYS     2432 85-08-23  9:29p
COMMAND  COM     6656 85-09-02 10:10p
AUTOEXEC BAT       77 87-02-12  3:58a
M80      COM    20480 85-07-11  6:46p
L80      COM    10752 85-07-11  7:01p
                 :
```

図 2.27　ドライブ B から DIR コマンドを実行

結果を見るとドライブBのディレクトリが表示されています．現在はカレント・ドライブがドライブBになっているので，もしドライブAのDIRを表示させたければ，**図 2.28** のように，きちんとドライブ名を指定しなくてはなりません．

```
B>dir a: ⏎
MSXDOS   SYS     2432 85-08-23  9:29p
COMMAND  COM     6656 85-09-02 10:10p
        2 files    719872 bytes free
B>
```

図 2.28　ドライブ B からドライブ A のディレクトリを表示

再びドライブAに戻ってみましょう(**図 2.29**)．

```
B>a: ⏎
A> ……………カレント・ドライブがドライブAになった.
```

図 2.29　カレント・ドライブを A に変更

このように，カレント・ドライブを変更することによってMSX-DOSが直接管理するドライブを変えることができます．

このカレント・ドライブの変更をうまく使えば，仕事の内容をドライブごとに分けることによって，ディスクの分割管理がしやすくなります．

では，話をDIRコマンドに戻しましょう．DIRコマンドには／Pと／Wの2つの**スイッチ**(コマンドに付いている拡張機能を使うか使わないかの指定)があります．

まず／Pから説明します．このスイッチは**ページ・モード**と呼ばれ，指定すると，ディレクトリ情報を1画面分表示したところで表示を中断します．任意のキーを押せば表示を再開します．**図2.30**に実行例を示します．

```
A>dir /p b:
MSXDOS   SYS    2432 85-08-23  9:29p
COMMAND  COM    6656 85-09-02 10:10p
AUTOEXEC BAT      77 87-02-12  3:58a
M80      COM   20480 85-07-11  6:46p
L80      COM   10752 85-07-11  7:01p
CREF80   COM    3968 85-04-01 10:22p
                              10:26p
CAL      COM    8832 87-03-04
CALC     COM   11776 87-03-04  1:09p
CHKDSK   COM    7168 87-02-16 12:47p
CLS      COM    6400 87-03-04  1:09p
DISKCOPY COM   13056 87-02-16  1:01p
DUMP     COM   10240 87-03-04  1:09p
ECHO     COM    6144 87-03-04  1:10p
EXPAND   COM    9216 87-03-04  1:10p
GREP     COM   11008 87-03-04  1:47p
Strike a key when ready
HEAD     COM    9472 87-03-04  1:11p
HELP     COM    6272 87-03-04  1:11p
KEY      COM    8192 87-03-04  1:11p
LIST     COM    9344 87-03-05 10:22a
LS       COM   11008 87-03-04  1:12p
MENU     COM   13696 87-03-04  1:18p
MORE     COM    9472 87-03-04  1:13p
                 87-03-04  1:13p
                           1:14p
```

1画面分表示したところで表示を中断．任意のキーで再開．

図2.30　ページ・モード

また，／W スイッチは**ワイド・ディスプレイ・モード**と呼ばれるもので，指定するとファイル名だけを１行にできる限り詰め込んで表示します．ディレクトリ情報の細かい点は気にしないで，どのような名前のファイルがあるかを確認する場合には便利な機能です．**図 2.31** を見てください．

```
A>dir /w b:⏎
MSXDOS   SYS    COMMAND  COM    AUTOEXEC BAT    M80      COM    L80      COM
CREF80   COM    LIB80    COM    MED      COM    MED      MES    MED      HLP
BEEP     COM    BIO      COM    BODY     COM    BSAVE    COM    CAL      COM
CALC     COM    CHKDSK   COM    CLS      COM    DISKCOPY COM    DUMP     COM
ECHO     COM    EXPAND   COM    GREP     COM    HEAD     COM    HELP     COM
KEY      COM    LIST     COM    LS       COM    MENU     COM    MORE     COM
PATCH    COM    SLEEP    COM    SORT     COM    TAIL     COM    TR       COM
UNIQ     COM    VIEW     COM    WC       COM    WELCOME  DOC    SAMPLE1  DOC
SAMPLE2  DOC    SAMPLE3  DOC    SEARCH   BAT    SEARCH   MAC    SEARCH   BAS
        45 files      5120 bytes free
A>
```

図 2.31　ワイド・ディスプレイ・モード

また，スイッチはファイル名のまえに付けても後ろに付けてもかまいません．つまり，〝DIR ／P B:〞と〝DIR B: ／P〞は同じ働きをします．

これらのスイッチは，ファイルの数が少ないときのはあまり使うメリットがないのですが，ファイルの数が増えてきて，ディレクトリが１画面に収まらなくなったときなど非常に便利な機能です．

さらに，DIR コマンドには**ワイルド・カード**と呼ばれる便利な機能が使えるのですが，これについては，3 章でファイルとファイル名についてきちんと説明したあとで取り上げることにしましょう．

2 5 コマンドの実行とコントロール・キーの操作

MSX-DOSでは，特別な意味を持つキーがいくつかありますので，説明しておきましょう．

MSX-DOSを使っていると，動作中のコマンドを中止したくなることがあります．たとえば，コマンド名を間違えて実行してしまったり，ディスクを間違えてフォーマットしそうになったときなどです．このような場合には，[CTRL]＋[C]を押します．すると，DOSは実行中のコマンドを中止してコマンド・レベルに戻ります．**図2.32**ではDIRコマンドの実行を途中で中止しています．

```
A>dir ⏎
MSXDOS   SYS     2432 85-08-23  9:29p
COMMAND  COM     6656 85-09-02 10:10p
AUTOEXEC BAT       77 87-02-12  3:58a
M80      COM    20480 85-07-11  6:46p
L80      COM    10752 85-07-11  7:01p
CREF80   COM     3968 85-04-01 10:22p
LIB80    COM     4736 85-04-01 10:26p
MED      COM^C ·············· [CTRL]+[C]を押す．
A>
```

図2.32　コマンドの実行中止

また，[CTRL]＋[P]を押すことにより，ディスプレイに表示される文字を同時にプリンタにも表示することができます．たとえば，ディスクの中にどのようなファイルがはいっているかを記録してきたい場合などに利用できます．**図2.33**のように，[CTRL]＋[P]を押したあと，DIRコマンドを実行すれば，結果は**図2.34**のようにプリンタに出力されます．この，ディスプレイの

内容をプリンタに表示する機能を中止したい場合には，CTRL + N を押します．

```
A>dir *.doc ↵ ··············· CTRL + P を押してから実行.
WELCOME  DOC     372 87-02-12  3:01p ┐
SAMPLE1  DOC      25 87-02-11  8:47p │
SAMPLE2  DOC      97 87-02-11  8:47p ├ プリンタにも出力される.
SAMPLE3  DOC      61 87-02-11  8:48p │
       4 files    5120 bytes free    ┘
A>dir *.mac ↵ ··············· CTRL + N を押してから実行.
SEARCH   MAC    2048 87-01-16  7:28p ┐ プリンタには出力されない.
       1 file    5120 bytes free     ┘
A>
```

図 2.33　ディスプレイへの出力

```
dir *.doc
WELCOME   DOC       372 87-02-12   3:01p
SAMPLE1   DOC        25 87-02-11   8:47p
SAMPLE2   DOC        97 87-02-11   8:47p
SAMPLE3   DOC        61 87-02-11   8:48p
         4 files      5120 bytes free
```

図 2.34　プリンタへの出力

ディスプレイへの表示を一時停止したい場合には，CTRL + S を押します．一度に大量の文章が表示されて読みきれないときにはこの機能が便利です．DIR コマンドでは，ページ・モードという機能がありましたが，こちらは任意の時点で表示を停止できるという点がすぐれています．また，画面に表示するコマンドであれば，3 章で紹介する TYPE コマンドなどにも使うことができます．一時停止を解除するには任意のキーを押します．**図 2.35** では DIR コマンドの実行を一時停止させています．

```
A>dir *.com
COMMAND  COM     6656 85-09-02 10:10p
M80      COM    20480 85-07-11  6:46p
L80      COM    10752 85-07-11  7:01p
CREF80   COM     3968 85-04-01 10:22p
LIB80    COM     4736 85-04-01 10:26p
MED      COM    15360 87-03-03 11:47p
BEEP     COM     5632 87-03-04  1:07p
BIO      COM    10240 87-03-04  1:07p
BODY     COM    10112 87-03-04  1:08p
BSAVE    COM     8832 87-03-04  1:08p
CAL      COM     8832 87-03-04  1:08p
CALC     COM    11776 87-03-04  1:09p
CHKDSK   COM     7168 87-02-16 12:47p
CLS      COM     6400 87-03-04█ ……… CTRL + S を押す.
                                   任意のキーを押すと表示を再開する.
```

図 2.35　コマンドの一時停止

以上，特殊キーの機能を簡単に紹介しましたが，最後に**表 2.2** にまとめておきます．MSX-DOS を使っていくうえで，これらの機能は大変便利なものです．ぜひ，使いこなしてください．

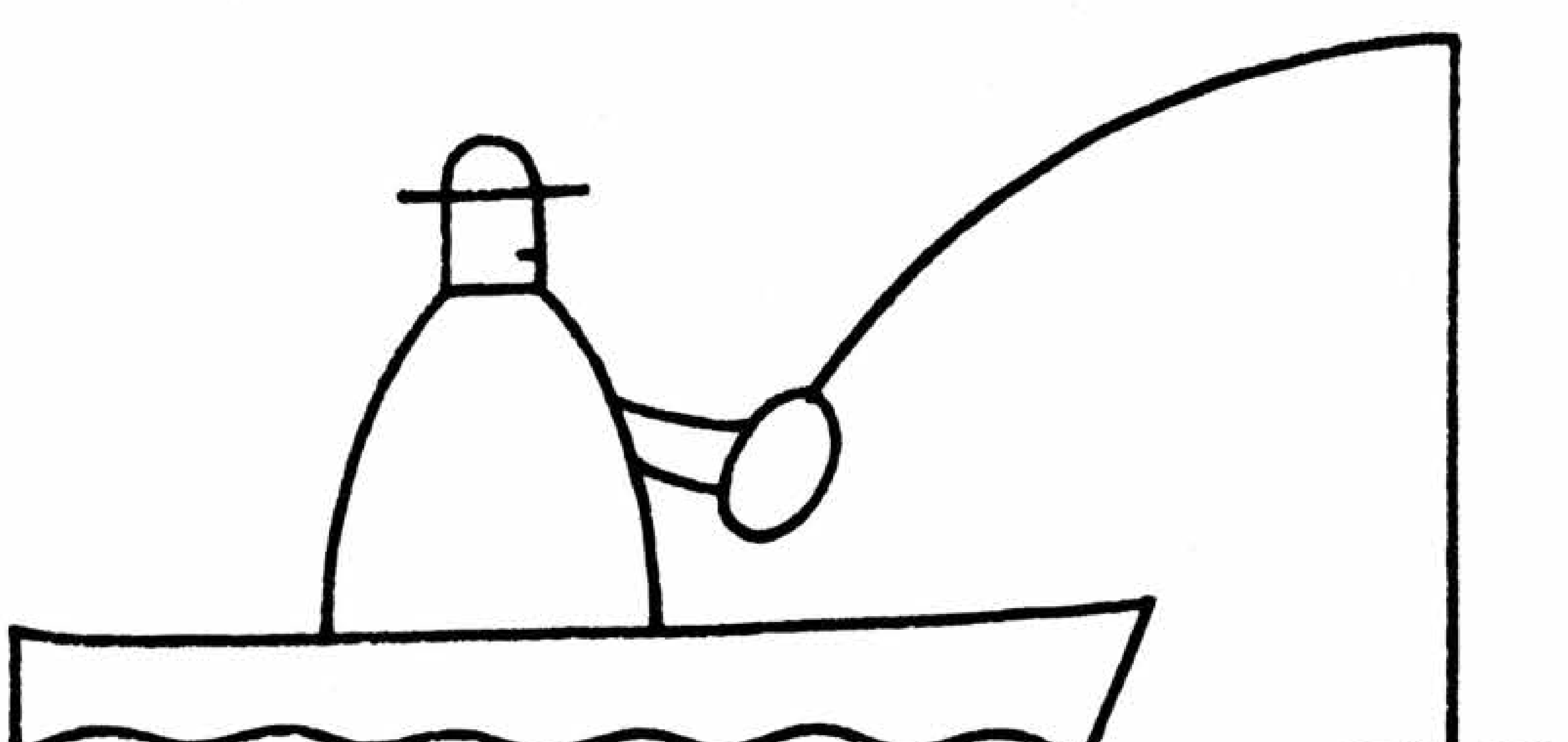
機　能	特殊キー	内　　　容
中止	CTRL + C	現在動作中のコマンドやプログラムの実行を中止します．
印刷	CTRL + P	ディスプレイへの表示をプリンタへも出力します
印刷取り消し	CTRL + N	CTRL + P の動作を取り消します
表示停止	CTRL + S	ディスプレイへの表示を一時停止します．任意のキーを押すと表示は再開します

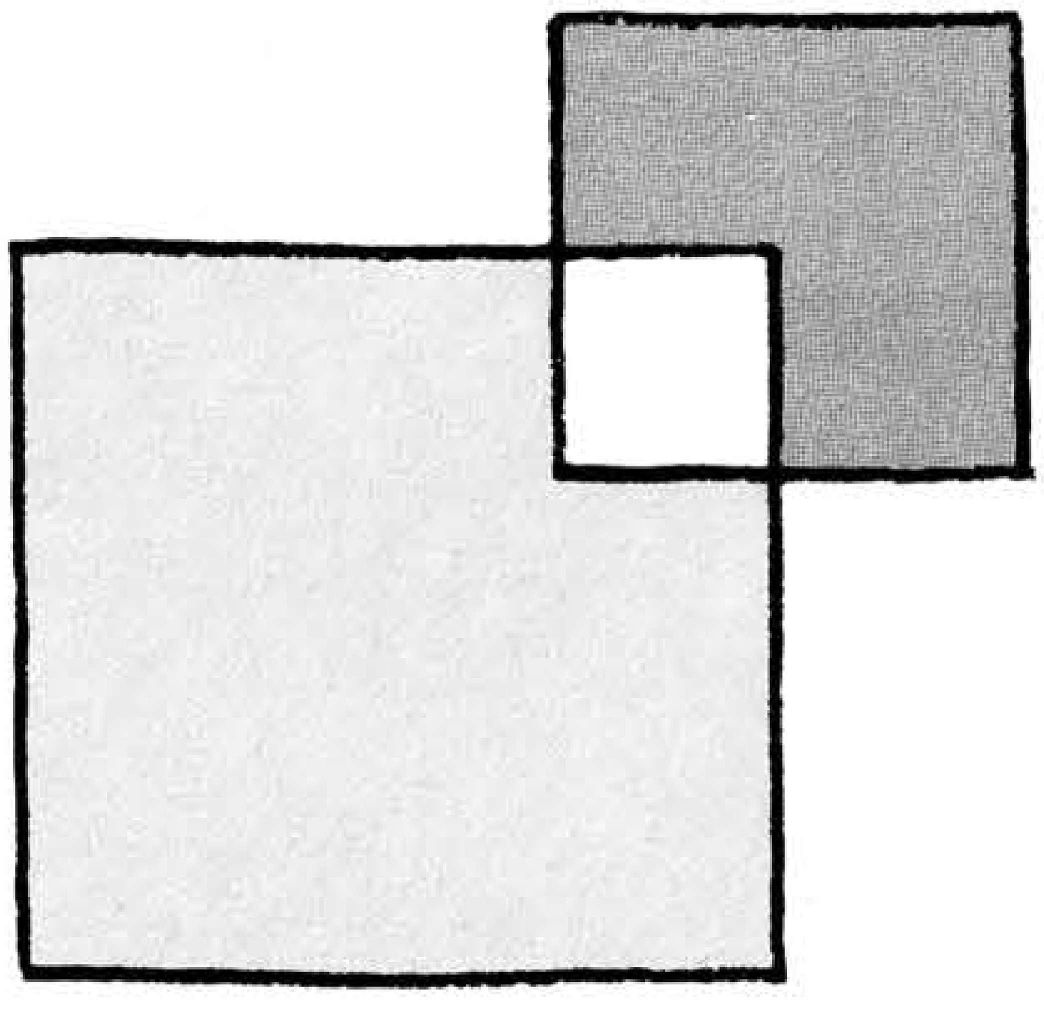
表 2.2　特殊キー一覧表

3章

ファイルの いろいろな処理

―ディスク上のファイルに対するコマンド操作―





3 1 ファイルとは

前章まででも，ファイルという言葉は使ってきました．いままでは漠然とディスクに記録されているデータのことをファイルと呼んできましたが，ここではファイルという重要な概念を，もう少し具体的な形で説明することにしましょう．

ファイルというのはなかなか抽象的な概念で説明しにくいのですが，ひとことでいえば，MSX-DOSが管理している，あるデータの集まりのことです．具体的な例をあげると，ディスク上に記録されているBASICのプログラム，ワープロの文書，グラフィック画面のデータなどがファイルに相当します．さらに，MSX-DOSでは，ディスク上のファイル以外にもいくつか特殊なファイルがあるのですが，それらの特殊なファイルについては4章でまとめて取り扱うことにして，ここでは，このディスク上のファイルについて，ファイル名の付け方，種類などに論点を絞って見ていくことにしましょう．

ファイル名の付け方

1枚のディスク上には，複数のファイルが存在しているのが普通ですから，その中から特定のファイルを識別するためには，ファイルにそれぞれ名前を付けておく必要があります．これは，人間のためにだけでなく，MSX-DOSにとっても必要なのです．いままでに出てきたファイル名としては，MSXDOS.SYSとCOMMAND.COMがあります．これら2例を見ると，ファイル名はピリオドで分けられる2つの部分からなることがわかります．ピリオドよりまえの部分は，**メイン・ファイル名**(またはプライマリ・ネーム，ベース・ネーム)と呼ばれ8文字以内の文字からなります．ピリオドの後ろに付けられる部分は，**エクステンション・ファイル名**(**拡張子**)といい，3文字以内でなくてはなりません．

ファイル名は，これら2つの部分から構成されますが，ピリオド以降の部分，つまり拡張子は必要なければ省略してもかまいません．

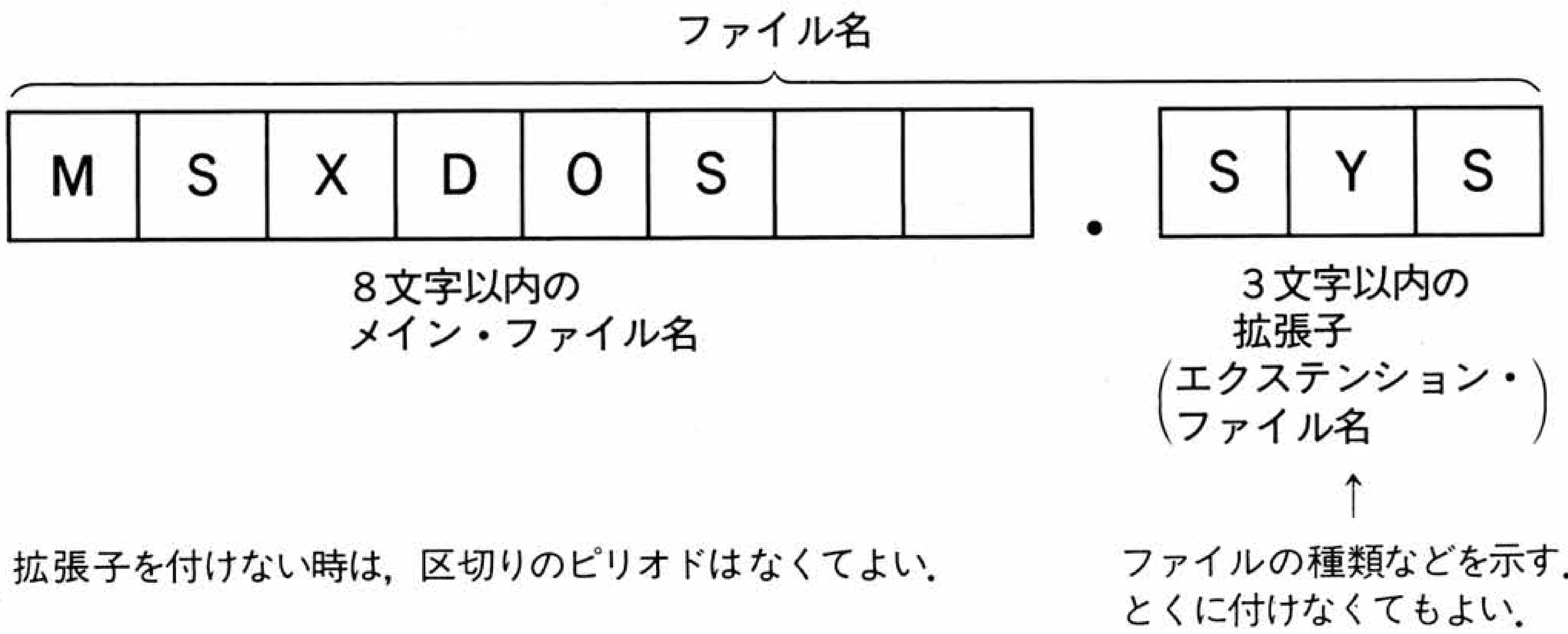


図 3.1　ファイル名

ファイル名に使える文字はなんでもよいというわけではなく，以下の文字の中から選ばなくてはなりません．

A～Z　0～9　$　&　#　@　!　%　'　(　)　－　{　}　～
ひらがな　カタカナ

アプリケーション・ソフトウェアによっては，上記の文字のうちでも特殊記号などの使用を認めていない場合があるので，英数字を使うのが無難です．また，文字の大文字，小文字は MSX-DOS では区別せず，すべて大文字に変換して扱います．つまり，〝mSx.DoS〟や〝Msx.dOS〟などは同じファイル名〝MSX.DOS〟として取り扱われるのです．

拡張子はファイルの種類を表すのに使います．これは，メイン・ファイル名と違って絶対に必要ということはありませんし，自分で好きな名前を付けてかまわないのですが，いくつかの拡張子は MSX-DOS で使い道が決まっていて特定の用途に使われます．これらの特別な意味を持つ拡張子は勝手に他の拡張子に変えて使うことはできません．**表 3.1** に MSX-DOS で用途が決まっている拡張子をあげておきます．

拡張子	ファイルの種類	例
.BAT	バッチ・ファイル	AUTOEXEC.BAT
.COM	コマンド・ファイル	COMMAND.COM
.SYS	システム・ファイル	MSXDOS.SYS

表 3.1 MSX-DOS で用途が決まっている拡張子

拡張子が BAT のものは**バッチ・ファイル**と呼ばれ，5 章で詳しく説明します．SYS は MSXDOS.SYS に使われているように，MSX-DOS のシステムを構成するファイルの拡張子として予約されています．

一番重要なのは拡張子が COM のファイルです．これは MSX-DOS から直接実行可能なファイルを表します．つまり，このファイルは MSX-DOS の組み込みコマンド(DIR などのように，あらかじめ MSX-DOS に組み込まれているコマンド)と同じように使えることになります．たとえば，画面をクリアするプログラムを作成して，CLS.COM というファイルを作ったとしましょう．すると，コマンド・レベルから CLS と入力するだけで，CLS.COM が実行され，画面がクリアされることになります．これは，MSX-DOS に CLS という画面クリアのコマンドが増えたのと同じことです．このような，実行可能な COM ファイルも広い意味ではコマンドと呼びます．つまり，DOS 上ではコマンドとアプリケーション・プログラムとの間に明確な区別はありません．極端な例ですが，DOS 上で動くゲームがあれば，それも一種のコマンドになります．これらの COM ファイルの形で存在するコマンドを**外部コマンド**といいます．

DIR コマンドや本章でこれから説明するコマンドには，対応する COM ファイルがありません(DIR.COM などは存在しません)．それなのに実行されるのは，これらのコマンドは COMMAND.COM というプログラムに一括して含まれているからなのです．これら，COMMAND.COM に含まれているコマンドを**内部コマンド**と呼び，COM ファイルによって作られる外部コマンドと区別します．

いろいろなファイル名

MSX-DOSには，慣習として用途の決まっている拡張子というものがあります．**表3.2**を見てください．これらは先の3つの拡張子とは違って，絶対に変えてはならないというものではありません．しかし，多くのプログラムで共通に使われているため，ユーザーがむやみやたらと利用すると，ファイル名が重複し，プログラムが使用している管理ファイルを破壊する恐れもあるので，注意して使うべきです．

拡張子	ファイルの種類	拡張子	ファイルの種類
.ASM	アセンブラ・ソース・ファイル	.FOR	FORTRANソース・ファイル
.BAK	バックアップ・ファイル	.HEX	インテル・hex・オブジェクト・ファイル
.BAS	BASICプログラム・ファイル	.LIB	アセンブラ・ライブラリ・ファイル
.C	Cソース・ファイル	.M80	アセンブラ・ライブラリ・ファイル
.COB	COBOLソース・ファイル	.MAC	アセンブラ・ソース・ファイル
.CRF	クロスリファレンス・ファイル	.PAS	PASCALソース・ファイル
.DAT	データ・ファイル	.PLI	PL／Iソース・ファイル
.DOC	ドキュメント・ファイル	.PRN	リスティング・ファイル
.FIF	FORTHソース・ファイル	.REL	オブジェクト・ファイル

表3.2　慣習として使われる拡張子

MSX-DOSでは，1枚のディスク上に存在する複数個のファイルに，まったく同じファイル名を付けることはできませんが，異なるディスク上であれば，同じファイル名を付けても，なんら問題はありません．このため，ドライブAのディスクとドライブBのディスクに，まったく同じファイル名のファイルが含まれているという場合も考えられます．この場合，ドライブ名まで指定することによって，これらを区別することが可能です．指定の方法は次のようになります．

書式　〈d：〉〈ファイル名〉　(〈d：〉はドライブ名の指定)

この，〈ファイル名〉の直前にドライブ名を付けて表したものを**ファイル・スペック**といいます．具体的には，以下のように表します．

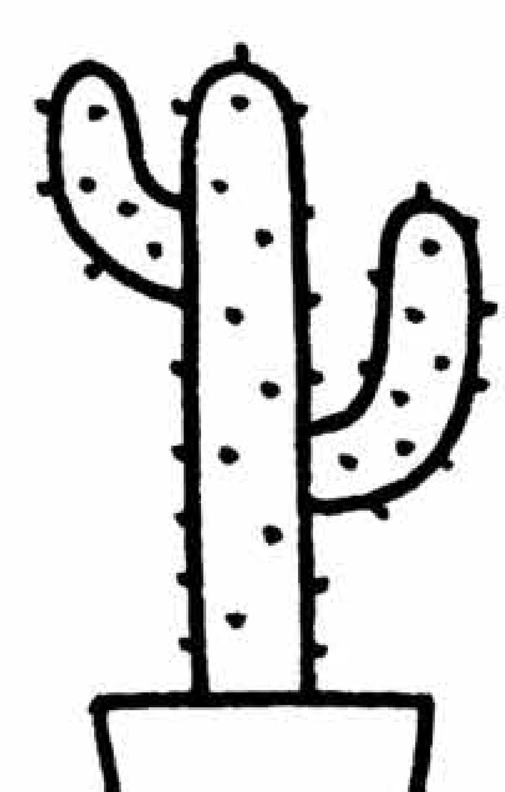
A:MSXDOS.SYS
B:MSXDOS.SYS
C:MSXDOS.SYS

MSX-DOSは，ドライブ名をとくに指定しなければ，カレント・ドライブ上のファイルとして認識するので，もし，カレント・ドライブ以外のファイルを指定したい場合には，かならずドライブ名も付けてファイル・スペックの形にする必要があります．

ファイル名についてあと1つだけ述べなくてはならないことがあります．メイン・ファイル名は基本的にどのような名前を付けてもよいのですが，例外として次の4つの名前は使えません．

AUX　CON　NUL　PRN

これらは，4章で説明する**デバイス・ファイル**の名前として予約されているため，勝手に使用するわけにはいきません．



3-2 ファイルのコピー

ファイルをコピーするには，**COPY コマンド**を使います．COPY は単にディスク上のファイルを複製するだけではなく，周辺装置とのデータの転送という機能も持った重要なコマンドです．周辺装置とのデータの転送については 4 章で詳しく説明するので，ここではディスク上でのファイルのコピーについて説明します．

また 3 章ではディスクに対するデータの書き込みを頻繁に行います．オリジナルのディスクを破壊しないように，2 章で作ったバックアップのシステム・ディスクを使うようにしてください．このディスクのライト・プロテクト・ノッチを書き込み許可の状態にして，ドライブ A に挿入します．カレント・ドライブが A であることを確認してから始めましょう．

ディスク上のファイルのコピー

COPY コマンドの書式は次のようになります．

書式　COPY　〈ファイル・スペック 1〉　〈ファイル・スペック 2〉

COPY コマンドは〈ファイル・スペック 1〉で表されるファイルを〈ファイル・スペック 2〉で表されるファイルにコピーします．実際にやってみましょう．コマンド・レベルから**図 3.2** と入力してください．

```
A>copy msxdos.sys test.1 ⏎
      1 file copied
A>
```

…………コマンド，〈ファイル・スペック 1〉，〈ファイル・スペック 2〉の間は，それぞれスペースで区切る．

↑ コピーに成功したファイルの数．

図 3.2　COPY コマンドの実行

このようにコピーが終了すると，コピーに成功したファイルの数を表示してコマンド・レベルに戻ります．この例ではドライブをとくに指定しなかったので，カレント・ドライブであるドライブAのMSXDOS.SYSというファイルを，ドライブAのTEST.1というファイルにコピーしたことになります(ドライブ名を指定しても結果は変わらない．つまり，"COPY　A:MSXDOS.SYS　A:TEST.1"としても同じ結果になる)．

ディレクトリを表示させてみましょう(**図3.3**)．

```
A>dir
MSXDOS   SYS     2432 85-08-23  9:29p ─┐
COMMAND  COM     6656 85-09-02 10:10p  │ コピーされた．
TEST     1       2432 85-08-23  9:29p ←┘
        3 files   716800 bytes free
A>
```

図3.3　コピーしたファイルの確認

図3.3を見るとTEST.1というファイルが増えています．ファイルをコピーすると，その内容だけでなく，作成された日付，時刻もコピーされます．そのため，TEST.1の内容はMSXDOS.SYSとまったく同じですが，ファイル名が違うのでMSX-DOSは異なるファイルとして認識します．

次にCOMMAND.COMをコピーしてみましょう．**図3.4**のように入力してください．

```
A>copy command.com test.1 ⏎
        1 file copied
A>dir ⏎
MSXDOS   SYS     2432 85-08-23  9:29p
COMMAND  COM     6656 85-09-02 10:10p ─┐
TEST     1       6656 85-09-02 10:10p ←┘ サイズが同じ．
        3 files   712704 bytes free
A>
```

図3.4　すでに存在しているファイルにコピー

ディレクトリを表示させるとわかるように，今度はファイルは増えていません．TEST.1のサイズを見るとCOMMAND.COMと同じであることがわかります．実際には，さきほどできたMSXDOS.SYSと同じ内容のTEST.1は消え，COMMAND.COMと同じ内容のTEST.1ができています．

このようにファイルをコピーする際に，コピー先に同じファイル名のファイルがあると，そのコピー先のファイルは消えてしまいます．ですから，不用意にCOPYコマンドを使うと，大切なファイルを消してしまう恐れがあります．DIRコマンドなどで確認してからCOPYコマンドを使うようにした方がよいでしょう．

では，次に新しいディスクを用意して，フォーマットしてください．フォーマットの方法については，すでに2章で説明したとおりです．このディスクを使ってDOSを起動させる必要はないので，システムはコピーしなくてけっこうです．

フォーマットがすんだら，この新しいデータ・ディスクをドライブBに入れてください(仮想ドライブ機能を使っている人は**図3.5**のコマンドを実行してから，そのあとに表示されるメッセージに従ってディスクを差し替えてください)．では，図3.5のコマンドを試してみてください．

```
A>copy test.1 b: ⏎ ……………ドライブBへのコピー．
        1 file copied
A>dir b: ⏎ ……………ドライブBのディレクトリを表示．
TEST     1        6656 85-09-02 10:10p ……………コピーされている．
        1 file    722944 bytes free
A>
```

図3.5　ドライブBへのコピー

ドライブBのディスクにはTEST.1というファイルができています．このようにコピー先にドライブ名だけを指定すると，コピー先のディスク上にコピー元と同じ名前のファイルを作ります．

もちろん，コピー先にファイル・スペックの形でファイル名を指定することも可能です．**図3.6**のように入力します．

```
A>copy test.1 b:sample.2 ⏎
        1 file copied
A>dir b: ⏎
TEST      1        6656 85-09-02 10:10p
SAMPLE    2        6656 85-09-02 10:10p ………ドライブBに指定したファイル名でコピーされている.
        2 files   715776 bytes free
A>
```

図 3.6 ドライブ B へのファイル名を指定したコピー

では，コピー元とコピー先を同じファイルにしたらどうなるでしょうか？

```
A>copy test.1 test.1 ⏎
File cannot be copied onto itself ……………エラーメッセージ
        0 files copied
A>      ↑――――ファイルはコピーされなかった.
```

図 3.7 コピー元とコピー先に同じファイル名を指定

図 3.7 のように，エラーメッセージが表示され，ファイルのコピーに失敗してしまいました．つまり，コピー元とコピー先は違うファイル・スペックでなくてはなりません．たとえファイル名は同じでも，**図 3.8** ようなことは可能です．

```
A>copy msxdos.sys b:msxdos.sys ⏎
        1 file copied
A>      ↑――――ファイルはコピーされている.
```

図 3.8 ファイル・スペックを変える

この場合，カレント・ドライブはドライブAですからコピー元は〝A:MSXDOS.SYS〟であり，コピー先は〝B:MSXDOS.SYS〟なのでファイル・スペックが違います．そのためコピーは成功します（この場合，コピー先には〝B:〟とだけ指定するのが普通です）．

また，**図 3.9** のような使い方も可能です．

```
A>copy b:sample.2 ↵ ……………〈ファイル・スペック 2〉が省略されている.
      1 file copied
A>dir sample.2 ↵
SAMPLE   2      6656 85-09-02 10:10p ………カレント・ドライブに
      1 file   705536 bytes free         コピーされた.
A>
```

図 3.9　コピー先の省略

この例では〈ファイル・スペック 2〉は完全に省略されています．この場合も，カレント・ドライブはドライブAですから，〈ファイル・スペック 2〉として〝A:〟と指定しているのと同じことになります．そのため，ドライブAにSAMPLE.2というファイルがコピーされているのです．

では，COPYコマンドの練習として以下の操作を行ってみてください．

①ドライブAのCOMANND.COMをドライブBにコピーする．
②ドライブAのMSXDOS.SYSをドライブBにTEST.2，SAMPLE.1，MSXDOS.BAK，MSX-DOS.SYSという4通りのファイル名でコピーする（COPYコマンドを4回使います）．
③ドライブAのCOMMAND.COMをドライブBにMSXCOM.1，COMMAND.BAKというファイル名でコピーする．

これらを正しく実行できたら，いままでコピーしたファイルと合わせて，ドライブBのディレクトリは**図 3.10** のようになるはずです．

```
A>dir b: ⏎
TEST      1       6656 85-09-02 10:10p
SAMPLE    2       6656 85-09-02 10:10p
MSXDOS    SYS     2432 85-08-23  9:29p
COMMAND   COM     6656 85-09-02 10:10p  ……①
TEST      2       2432 85-08-23  9:29p
SAMPLE    1       2432 85-08-23  9:29p
MSXDOS    BAK     2432 85-08-23  9:29p  ……②
MSX-DOS   SYS     2432 85-08-23  9:29p
MSXCOM    1       6656 85-09-02 10:10p
COMMAND   BAK     6656 85-09-02 10:10p  ……③
       10 files   678912 bytes free
A>
```

図 3.10 COPY コマンドの練習後のドライブ B のディレクトリ

COPY 命令の最後に，仮想ドライブ機能使用時の大容量ファイルのコピーについて説明しておきます．これまで登場した，MSXDOS.SYS や COMMAND.COM のような小さいファイルのコピーを行う際には関係ありませんが，ある程度大きいプログラムのコピーを，仮想ドライブ機能で行おうとすると，ディスクの入れ替えが一度ではすまないことがあります．そのような場合には，ドライブ B へのデータの書き込みが終了したのちに，ディスク・ドライブがドライブ A に切り替わり，**図 3.11** のようなメッセージが表示されます．

```
Insert diskette for drive A:
and strike a key when ready
```

このメッセージが表示された場合には，もう 1 度ソース・ディスクに差し替えて，任意のキーを押す．

図 3.11 ソース・ディスクに差し替えるメッセージ

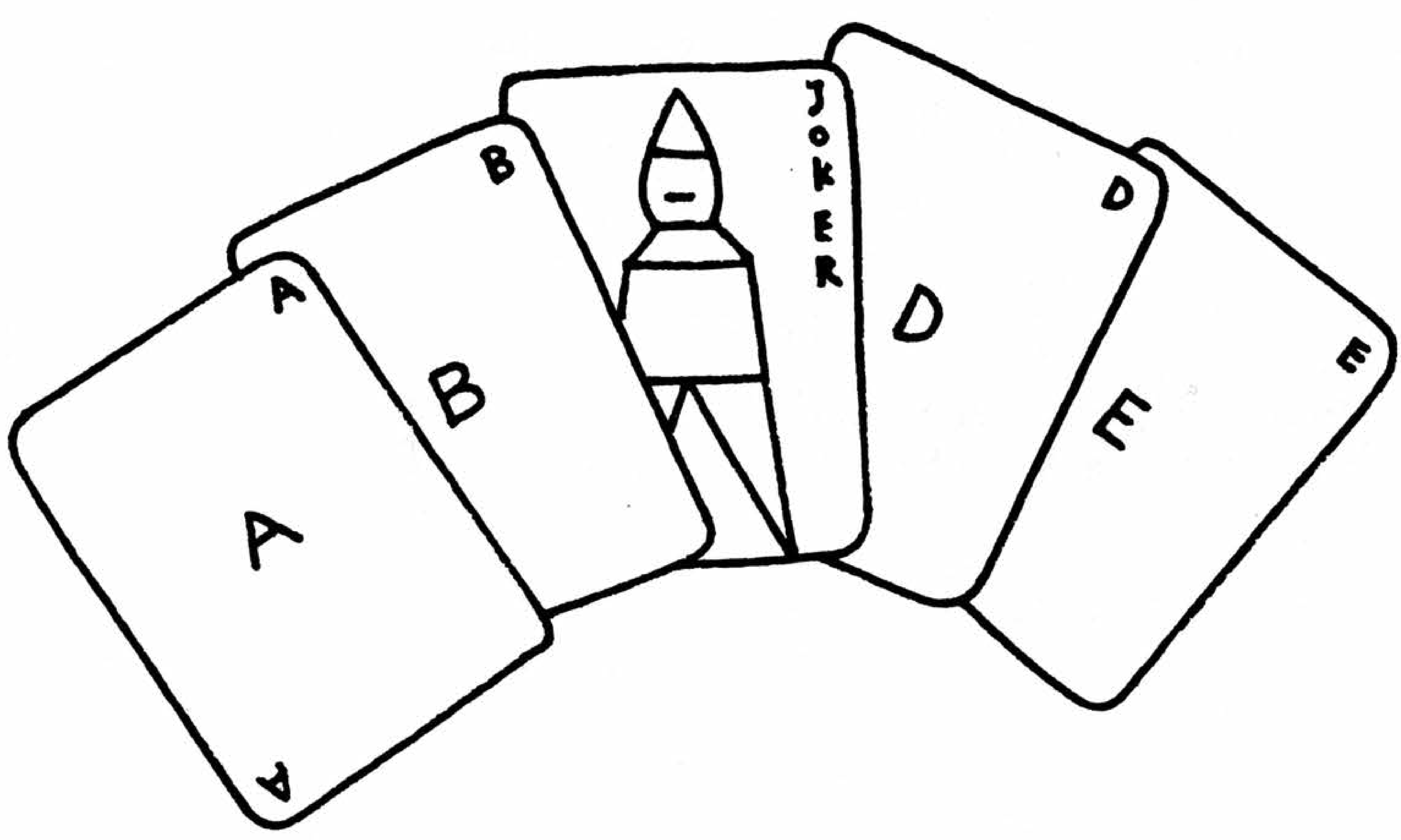
このメッセージが表示された場合には，ドライブ A にはいっているはずのソース・ディスクを再び挿入して，何かキーを押します．すると，ドライブ A からデータのつづきを読み込みます．そして，最初に表示されたディスク差

し替えメッセージが再び表示され，ディスク・ドライブがBドライブに切り替わるので，デスティネーション・ディスク差し替えて，さきほどと同じように操作します．

コピーするファイルが非常に大きいと，この操作を繰り返す回数も増えます．その途中で，ドライブAならソース・ディスクを，ドライブBならデスティネーション・ディスクを間違わないように挿入してください．

ワイルドカード

前章のDIRコマンドのところで約束したように，ここでワイルドカードの説明をすることにしましょう．ワイルドカードとはファイル名を指定する際に，任意の文字(列)の代わりに使える文字のことで，**クェスチョンマーク**(？)と**アスタリスク**(＊)の2種類があり，目的によって使い分けます．これらは，DIRコマンドだけではなく，COPYなどのコマンドに対しても使うことができる応用範囲の広いものです．



●クエスチョンマーク(?)

これは，任意の1文字の代わりに使えます．たとえば，次のように使います．前節でコピーの練習に使ったディスク(ドライブBに挿入していたもの)をドライブAに入れてから，**図3.12**のように入力してみてください．

```
A>dir b: ⏎
TEST      1       6656 85-09-02 10:10p
SAMPLE    2       6656 85-09-02 10:10p
MSXDOS    SYS     2432 85-08-23  9:29p
COMMAND   COM     6656 85-09-02 10:10p
TEST      2       2432 85-08-23  9:29p
SAMPLE    1       2432 85-08-23  9:29p
MSXDOS    BAK     2432 85-08-23  9:29p
MSX-DOS   SYS     2432 85-08-23  9:29p
MSXCOM    1       6656 85-09-02 10:10p
COMMAND   BAK     6656 85-09-02 10:10p
      10 files    678912 bytes free
A>dir test.? ⏎
TEST      1       6656 85-09-02 10:10p } 該当するファイル
TEST      2       2432 85-08-23  9:29p }
       2 files    678912 bytes free
A>
```

図3.12　ワイルドカード〝?〟の使用例

図3.12の例では，拡張子を〝?〟にしています．つまり，このコマンドの意味は「メイン・ファイル名がTESTで，拡張子が1文字のものについてディレクトリを表示しろ」となります．

また，〝?〟は複数でも使用可能です(**図3.13**)．

```
A>dir msx???.bak ⏎
MSXDOS    BAK     2432 85-08-23  9:29p ……………該当するファイル
       1 file     678912 bytes free
A>
```

図3.13　ワイルドカード〝?〟を複数使用

図3.13のコマンドは,「メイン・ファイル名がMSXで始まり，その後3文字は何でもいいが, 拡張子はBAKであるもののディレクトリを表示しろ」という意味になります.

また，ワイルドカードによって示された条件と，ファイル名が適合することを，ファイル名がマッチするといいます.

次のようなファイル名を考えてみましょう.

① ABCDEF.ONE
② ABCABC.TWO
③ AACCEE.ONE
④ AAADEF.1
⑤ DEFDEF.2

この中で,〝ABC???.???〞とマッチするのはどれでしょうか？　もうおわかりですね. ①と②です. では,〝A?C?E?.???〞ならどうでしょう？　①と③ですね. 同様に〝???DEF.?〞なら④と⑤にマッチします.

このように, ワイルド・カードを使うことによって, 具体的なファイル名を与えるよりも, もっと広い範囲でファイルを指し示すことが可能になります.

しかし，これだけではファイルの文字数がわからないと使えないので不便です. そこで〝*〞が必要となるのです.

●アスタリスク(*)

〝*〞は任意の文字数の文字列とマッチするワイルドカードです. では，さっそく使ってみましょう(**図3.14**).

```
A>dir *.sys ⏎
MSXDOS   SYS     2432 85-08-23   9:29p  } ファイル名がマッチ
MSX-DOS  SYS     2432 85-08-23   9:29p  } するもの.
       2 files    678912 bytes free
A>
```

図3.14　ワイルドカード〝*〞の使用例

図3.14は,「拡張子がSYSのものすべてのディレクトリを表示しろ」という意味のコマンドになります. 再び〝?〟のところで考えた5つのファイル名について考えてみましょう.

① ABCDEF.ONE
② ABCABC.TWO
③ AACCEE.ONE
④ AAADEF.1
⑤ DEFDEF.2

ここで,〝ABC *.*〟とマッチするのは①,②の2つになります. また⑤以外は,〝A *.*〟とマッチするわけです.

では,〝* C.*〟というようなワイルドカードとマッチするのはどれでしょうか? ちょっと考えると②とだけマッチしそうに思えます. しかし,実際にはすべてのファイルとマッチしてしまいます. 実際にMSXでやってみましょう(図3.15).

```
A>dir *c.*
TEST      1      6656 85-09-02 10:10p
SAMPLE    2      6656 85-09-02 10:10p
MSXDOS    SYS    2432 85-08-23  9:29p
COMMAND   COM    6656 85-09-02 10:10p
TEST      2      2432 85-08-23  9:29p
SAMPLE    1      2432 85-08-23  9:29p
MSXDOS    BAK    2432 85-08-23  9:29p
MSX-DOS   SYS    2432 85-08-23  9:29p
MSXCOM    1      6656 85-09-02 10:10p
COMMAND   BAK    6656 85-09-02 10:10p
       10 files   678912 bytes free
A>
```

(すべてのファイルとマッチしている.)

図3.15 〝* C.*〟とマッチするファイル

いままで，COPYコマンドの練習に使ってきたディスク上には，メイン・ファイル名がCで終わるようなファイルはないはずです．しかし，図3.15の結果を見るとすべてのファイルのディレクトリが表示されています．どうしてこのような結果になるのでしょうか？

これは，MSX-DOSのファイル名の管理方法と関係があります．**図3.16**を見てください．MSX-DOSでは，メイン・ファイル名は8文字まで，拡張子は3文字までと決まっています．もし，ファイル名がこの最大文字数より短い場合，MSX-DOSは空白を補充して記憶しておきます．

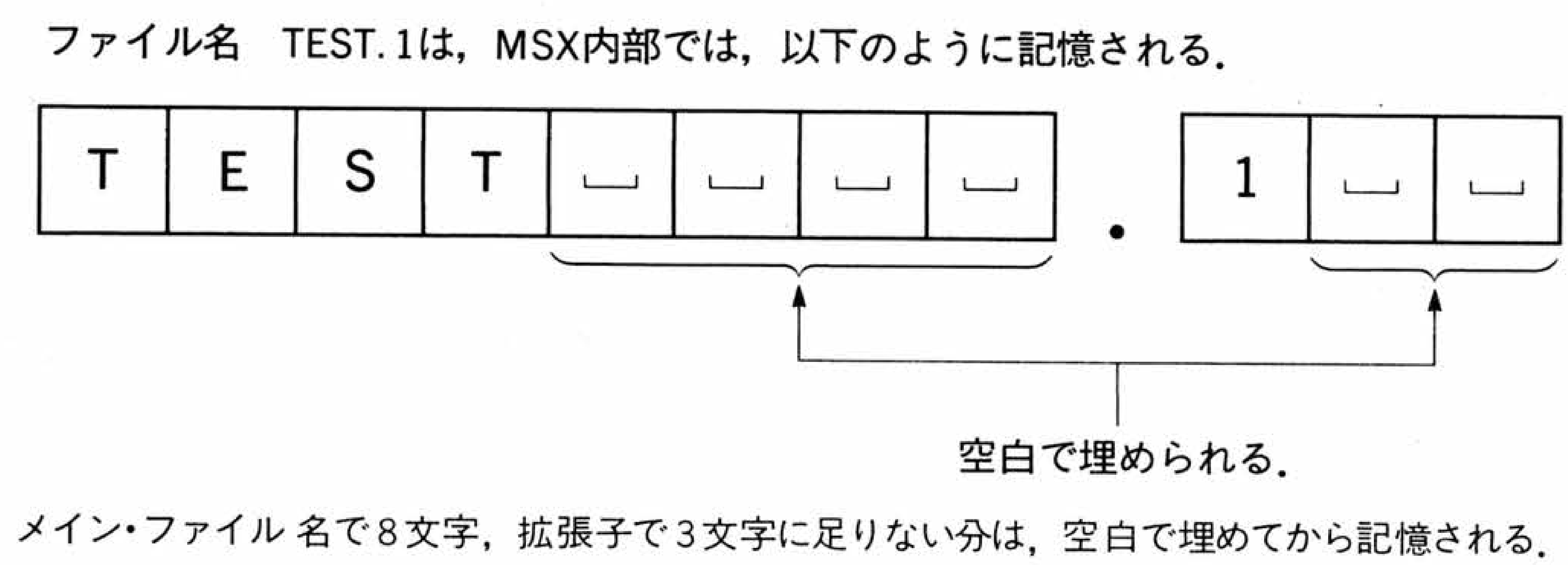


図3.16　ファイル名に空白を補充

ですから，ワイルドカード〝？？？？？？？？.？？？〟はすべてのファイルとマッチします．つまり，〝＊.＊〟と同じことです．

実際，〝＊〟が使われるとMSX-DOSはそれを〝？〟で置き替えてからファイル名とマッチさせていたのです．

具体的には，〝ABC＊.＊〟というワイルドカードはまず〝＊〟を8個の〝？〟で置き替えます．すると，〝ABC？？？？？？？？.？？？？？？？？〟となりますが，ファイル名の最大文字数を超えた分はカットされ〝ABC？？？？？.？？？〟となり，マッチするファイル名を探すのです．では，同じように〝＊C.＊〟を考えてみましょう．結果は，**図3.17**のようになります．

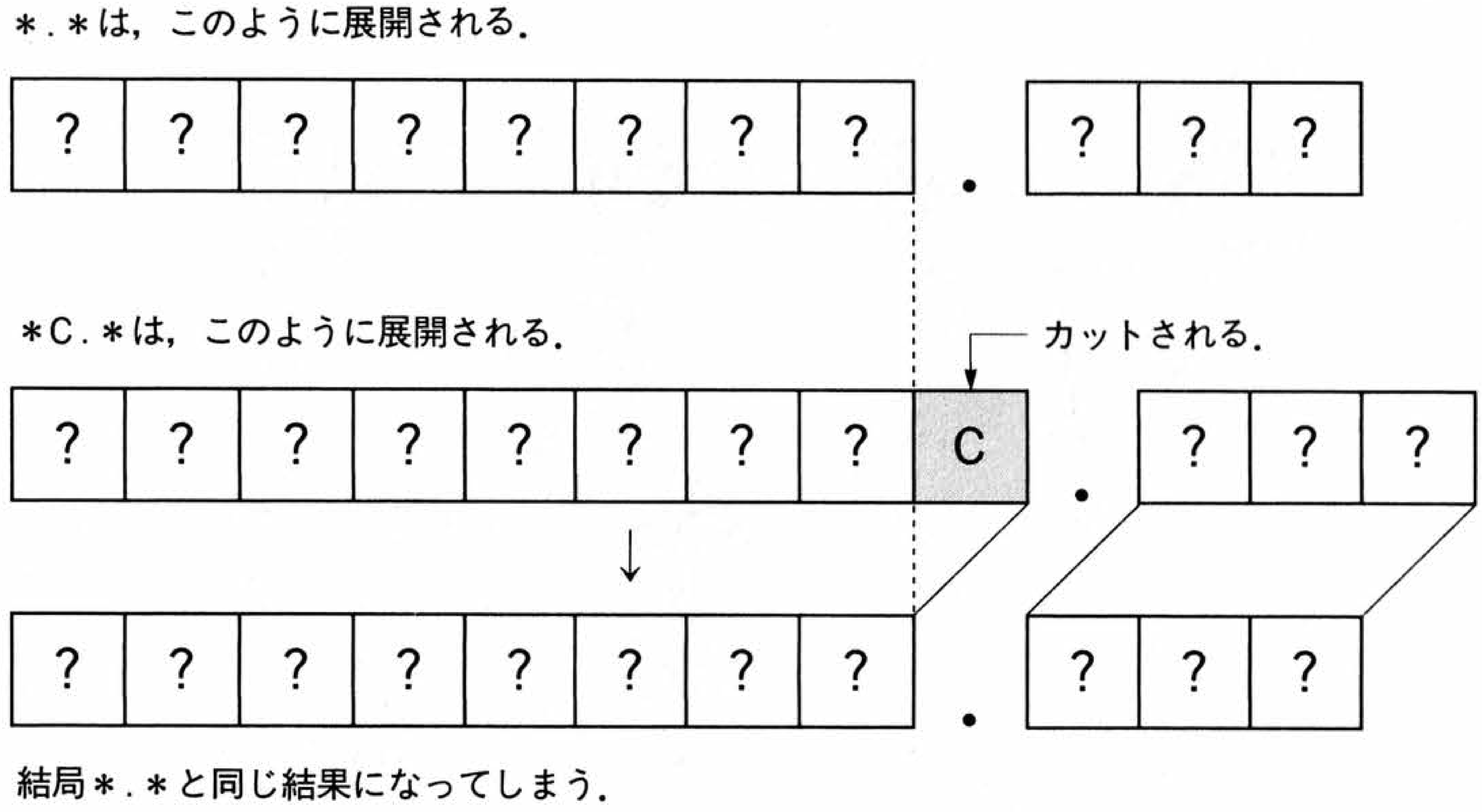
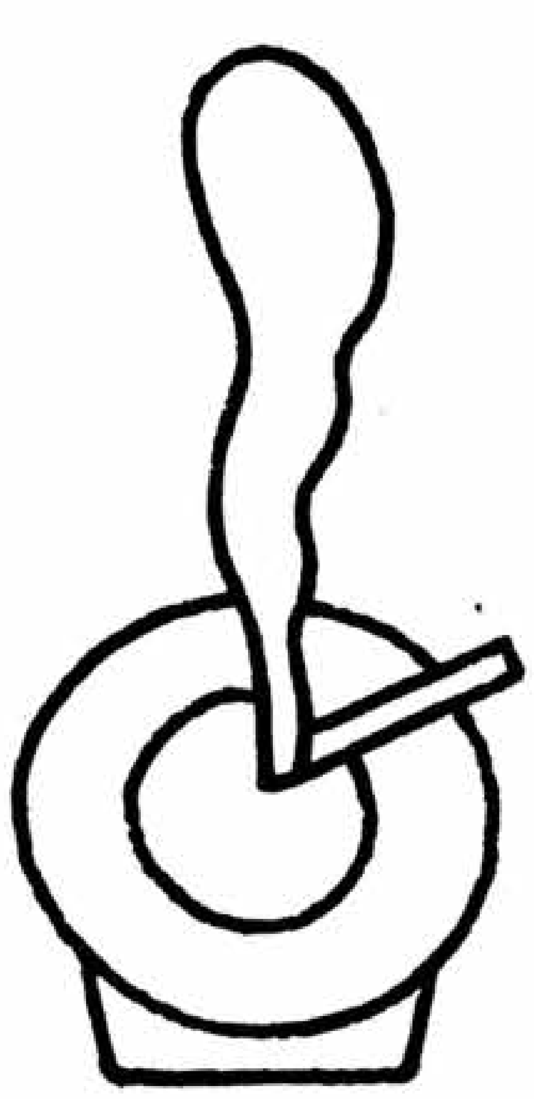


図 3.17 〝＊ C.＊〞の展開図

これは，〝＊.＊〞を展開した結果と同じです．このため，すべてのファイルとマッチしてしまったのです．

このように，〝＊〞の後ろに文字を指定しても無視されてしまいますから気を付けてください．



3 3 ファイル名の変更

これで，ファイルをコピーすることが可能となりましたが，この際に気を付けなくてはならないのは，コピー先のディスクに同じファイル名のファイルがあるとそのファイルは消えてしまうということでした．

もし，そのファイルを消したくなければ，あらかじめファイル名を変更しておかなくてはなりません．そのための命令が**REN(RENAME)コマンド**です．書式は次のようになります．

書式　**REN**　**〈ファイル・スペック〉**　**〈ファイル名〉**

このように指定すると，〈ファイル・スペック〉で示されるファイルは〈ファイル名〉で与えられるファイル名に変更されます．

また，このコマンドは"RENAME"とフルスペルを書いてもかまいませんが，普通は省略形の"REN"を使います．では，実際に使ってみましょう．

```
A>ren test.1 sample.3 ↵
A>dir *.? ↵
SAMPLE   3      6656 85-09-02 10:10p ……………ファイル名が変わった．
SAMPLE   2      6656 85-09-02 10:10p
TEST     2      2432 85-08-23  9:29p
SAMPLE   1      2432 85-08-23  9:29p
MSXCOM   1      6656 85-09-02 10:10p
       5 files   678912 bytes free
A>
```

図 3.18　REN コマンドの実行例

図3.18のように，ディレクトリを表示させてみれば，TEST.1がSAMPLE.3に変わっているのがわかります．ただし，変わっているのはファイル名だけでファイルの中身はまったく変わっていません．

また，RENコマンドではワイルドカードを使うことができます．これも実際に試してみましょう．**図3.19**のように入力してみてください．

```
A>ren ???-*.* ???!*.*
A>dir
SAMPLE   3       6656 85-09-02 10:10p
SAMPLE   2       6656 85-09-02 10:10p
MSXDOS   SYS     2432 85-08-23  9:29p
COMMAND  COM     6656 85-09-02 10:10p
TEST     2       2432 85-08-23  9:29p
SAMPLE   ONE     2432 85-08-23  9:29p
MSXDOS   BAK     2432 85-08-23  9:29p
MSX!DOS  SYS     2432 85-08-23  9:29p ……ファイル名が変わった．
MSXCOM   ONE     6656 85-09-02 10:10p
COMMAND  BAK     6656 85-09-02 10:10p
      10 files    678912 bytes free
A>
```

図3.19 RENコマンドでのワイルドカードの使用

図3.19を見ると，ファイル名の4文字目に〝-〟があるものを探して〝!〟で置き替えていることがわかります．

3 4 ファイルの作成と内容の表示

ファイルのコピー，ファイル名の変更などのコマンドを説明しましたが，まだ自分でファイルを作る方法を説明していません．

ひとくちにファイルを作るといっても，即実行可能なCOMファイルのようなものから，ワープロの文書データなどを記憶しておくファイルまでさまざまなものがあります．しかし，かりにCOMファイルを作るとしても，結局普通の文字を使って書いたプログラムをマシン語プログラムに変換しなければならないのですから，まずは文書データを記憶しておくファイルを作らなければなりません．

このような文書を記憶しておくファイルのことを**テキスト・ファイル**(**アスキー・ファイル**)といいます．このテキスト・ファイルを作り出すためのアプリケーション・ソフトウェアには，ワープロ（日本語MSX-Writeなど）やエディタ（MEDなど）と呼ばれるものがあります．これらを使えば，文書の編集・変更などは能率的に行うことができます．しかし，とりあえずテキスト・ファイルを作ってみるという私たちの目的には，これほど大がかりなものは必要ありません．

ファイルの作成

最も簡単にテキスト・ファイルを作るには，デバイス・ファイルCONというものを使いますが，この方法は次章で説明することにして，ここでは身近なDISK-BASICを使ってみることにしましょう．

DISK-BASICはMSX-DOSのアプリケーション・ソフトウェアの1つですから，DISK-BASICで作ったファイルをMSX-DOSで取り扱うことができます．

まず，DOSからBASICを起動してみます．コマンド・ラインから"BASIC"

と入力してください．このコマンドによってDISK-BASICが起動します(**図3.20**)．

```
A>basic ⏎
```

図 3.20 BASIC の起動

BASICが起動したら，**リスト 3.1** のプログラムを入力してください．

```
10 'random
20 OPEN "test1.doc" AS #1 LEN=1
30 FIELD #1, 1 AS A$
40 READ R$
50 IF R$ = "!!" THEN END
60 LSET A$ = R$
70 PUT #1
80 GOTO 40
90 DATA T,E,S,T,!!
```

リスト 3.1 ランダム・ファイル作成プログラム

リスト3.1のプログラムは，〝TEST1.DOC〟というファイル名のテキスト・ファイルを作り，DATA文で指定した文字を記録します．このプログラムを実行(RUN)させたら，もう1つ，**リスト 3.2** の BASIC のプログラムを入力してください．

```
10 'sequential
20 OPEN "test2.doc" FOR OUTPUT AS #1
30 R$ = "TEST"
40 PRINT #1, R$;
50 END
```

リスト 3.2 シーケンシャル・ファイル作成プログラム

このプログラムも実行させないとファイルが作られませんので，かならず実行させてください．このプログラムを使って作った"TEST2.DOC"がさきほどのリスト3.1のプログラムを使って作った"TEST1.DOC"とどこが違うかということについてはあとで触れます．

皆さんも，これらプログラムを参考にして何かのテキスト・ファイルを作ってみてください．ただし，ファイル名はまだ使っていない新しいものにしないとせっかく作ったファイルが消えてしまいます．気を付けてください．

さて，DOSに戻りましょう．"CALL　SYSTEM"と入力してください（図3.21）．

```
call system ⏎
A> ……………DOSに戻った．
```

図3.21　BASICの終了

DOSに戻ったらディレクトリを表示させてみましょう（図3.22）．

```
A>dir ⏎
SAMPLE   3        6656 85-09-02 10:10p
SAMPLE   2        6656 85-09-02 10:10p
MSXDOS   SYS      2432 85-08-23  9:29p
COMMAND  COM      6656 85-09-02 10:10p
TEST     2        2432 85-08-23  9:29p
SAMPLE   ONE      2432 85-08-23  9:29p
MSXDOS   BAK      2432 85-08-23  9:29p
MSX!DOS  SYS      2432 85-08-23  9:29p
MSXCOM   ONE      6656 85-09-02 10:10p
COMMAND  BAK      6656 85-09-02 10:10p
TEST1    DOC         4 87-12-24  4:05p
TEST2    DOC         5 87-12-24  4:07p  ……BASICのプログラムで
      12 files   676864 bytes free         作られたファイル
A>
```

図3.22　作成されたファイルの確認

図3.22のように，〝TEST1.DOC〟と〝TEST2.DOC〟が作られています．これらがさきほどBASICで書き込んだ内容になっているかどうか調べてみましょう．

テキスト・ファイルの内容を表示させるには**TYPEコマンド**を使います．書式は次のようになります．

書式　TYPE　**〈ファイル・スペック〉**

では，TYPEコマンドを使ってみましょう．**図3.23**のように入力してください．

```
A>type test1.doc ⏎
TEST
A>type test2.doc ⏎
TEST
A>
```
（指定したファイルの内容を表示．）

図3.23　TYPEコマンドの実行例

このようにBASICで作ったファイルに文章が記録されています．これは，BASICがMSX-DOSのアプリケーション・ソフトウェアの1つであるためです．

また，TYPEコマンドはテキスト・ファイルを表示するためのコマンドなので，COMMAND.COMなどのテキスト・ファイル以外のファイルの内容を表示しようとしても，正しく表示されません．**図3.24**はCOMMAND.COMをTYPEコマンドで表示した結果です．

```
*-58;CFIP¥♣うこそ｢ｬｵｸｽﾀﾉ!$';>Qfpsvyをぅゃしｴﾁﾄふまもられ        ●えこ ｲｵﾖﾇﾊﾏもたてにひほめゆれん"%(?Vcjm○けせ
A>ｱｯｴｷﾀﾃﾆﾉﾐﾘﾛﾞちなねひめゆ
```

図3.24　COMMAND.COMをTYPEコマンドで表示

TEST1.DOCとTEST2.DOCは，両者ともBASICで作ったテキスト・ファイルには，微妙な違いがあります．これは，テキスト・ファイルの本質に関わる違いなので，ここできちんと説明しておきましょう．

EOFについて

コンピュータ内部では，文字も数値データとして取り扱わなければなりません．そのため，すべての文字には，対応する文字コードというものが用意されています．この文字コードを一般的に取り決めたものに，ASCIIコードやJISコードといった規格があります．

MSXはJISコード準拠となっており，英数字は，1文字を1バイトのコード(00h～FFhまでの数値，数字のあとのhは16進数を表す)で表現していますから，本来はTEST1.DOCもTEST2.DOCも4文字なので4バイトのファイルになるはずです．ところが，図3.22を見ると文字数は両方とも4文字で同じなのに，TEST1.DOCよりTEST2.DOCの方が1バイトだけサイズが大きくなっています．

これは，TEST2.DOCの最後に，テキスト・ファイルの終わりを示す**EOF**(End of File)と呼ばれる1バイトのコードが付け加えられているためです．コンピュータは数値しか扱えませんから，このEOFも数値で表されなくてはなりません．EOFには，普通は文字として使わない1Ah(コントロール文字のCTRL-Zに当たる)が割り当てられています．EOFはテキスト・ファイルの最後を表すためのマークなので，テキスト・ファイルの最後にかならずなくてはなりません．TEST1.DOCはEOFがありませんから，不完全なテキスト・ファイルということになります．

前項で説明したTYPEコマンドは，テキスト・ファイルに対して行う命令ですから，ファイルの先頭から最初に出てきたEOFまでを表示します．ファイル中にデータが残っていてもEOFから先は表示しません．TEST1.DOCはEOFがありませんから，TYPEコマンドはファイルの最後まで表示します．一方，COMファイルはテキスト・ファイルではないので，ファイルの最後にEOFを付ける必要はありません．しかし，マシン語のプログラムは00hからFFhまで，すべてのコードを使って記述されているので，1Ahを使って

いる部分もあるはずです．そのため COMMAND.COM を TYPE コマンドで表示した際には，プログラム中の最初の 1Ah が EOF だと認識され，そこで表示が打ち切られていたのです(図 3.25)．

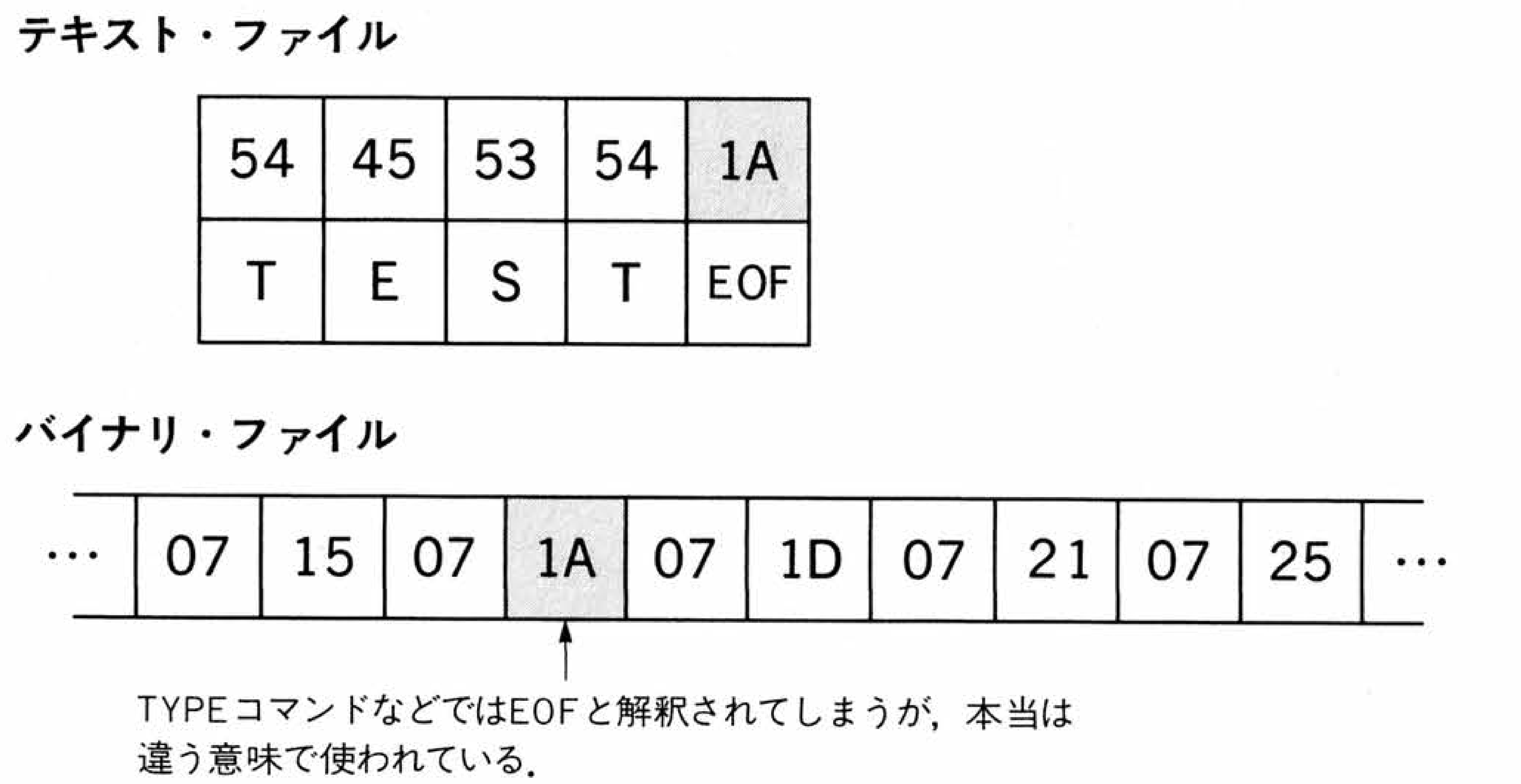


図 3.25　テキスト・ファイルにとっての 1Ah と，バイナリ・ファイルにとっての 1Ah

さて，TEST1.DOC と TEST2.DOC の違いが生じた理由は，リスト 3.1 のプログラムとリスト 3.2 のプログラムの違いにあります．BASIC でファイルを作る方法には，ランダム・ファイルとシーケンシャル・ファイルの 2 通りがあります．実は，リスト 3.1 のプログラムはランダム・ファイル，リスト 3.2 のプログラムはシーケンシャル・ファイルを使ってファイルを作っていたのです．

つまり，シーケンシャル・ファイルを使うと EOF の付いた完全なテキスト・ファイルができ，ランダム・ファイルを使うと EOF のない不完全なテキスト・ファイルができることになります．したがって，マシン語のプログラムのように，テキスト・ファイル以外のファイルを作る場合には，EOF は必要ないので，ランダム・ファイルで作りますが，テキスト・ファイルを作る場合にはシーケンシャル・ファイルで作ります．

次節でファイルの連結について説明しますが，この EOF の有無で方法が変わってきます．ここでは，文章を表すファイル，つまりテキスト・ファイルの最後には EOF が必要であり，それ以外のファイル(バイナリ・ファイルといいます)は EOF を必要としないということを覚えておいてください．

3 5 ファイルの連結

ファイルの連結について説明することにしましょう．ファイルのコピーと同様に COPY コマンドを使います．書式は次のようになります．

書式　COPY　〈ファイル・スペック 1〉＋〈ファイル・スペック 2〉＋……＋〈ファイル・スペック n〉　〈ファイル・スペック〉

連結したいもとのファイル(ソース)のファイル・スペックを〝＋〞でつなげ，並べて指定し，最後にそれらを連結して作りたいファイル(デスティネーション)のファイル・スペックを指定します．

実際にやってみましょう．図 3.26 のように入力してください．

```
A>copy test2.doc + test2.doc test3.doc
        1 file copied
A>
```

図 3.26　COPY コマンドによるファイルの連結

これで TEST2.DOC が 2 つ，つながったファイル TEST3.DOC ができあがりました．TYPE コマンドを使って TEST3.DOC の内容を確認してみましょう．図 3.27 を見てください．

```
A>type test3.doc
TESTTEST ……………連結されている.
A>
```

図 3.27　連結されたファイルの内容確認

ファイルの連結モード

TYPEコマンドはEOFまでを表示するので，TEST3.DOCにはEOFが最後に1つだけあるということになります(途中には，はいっていない).

このように，MSX-DOSはテキスト・ファイルに対してはEOFをいったん取ってから連結し，最後にEOFを1つ付けます．このように，EOFを自動的に調節してテキスト・ファイルにEOFが余分に付かないようにファイルを連結してくれるモードを**アスキー・モード(テキスト・モード)**といいます．これを図で示すと**図3.28**のようになります．

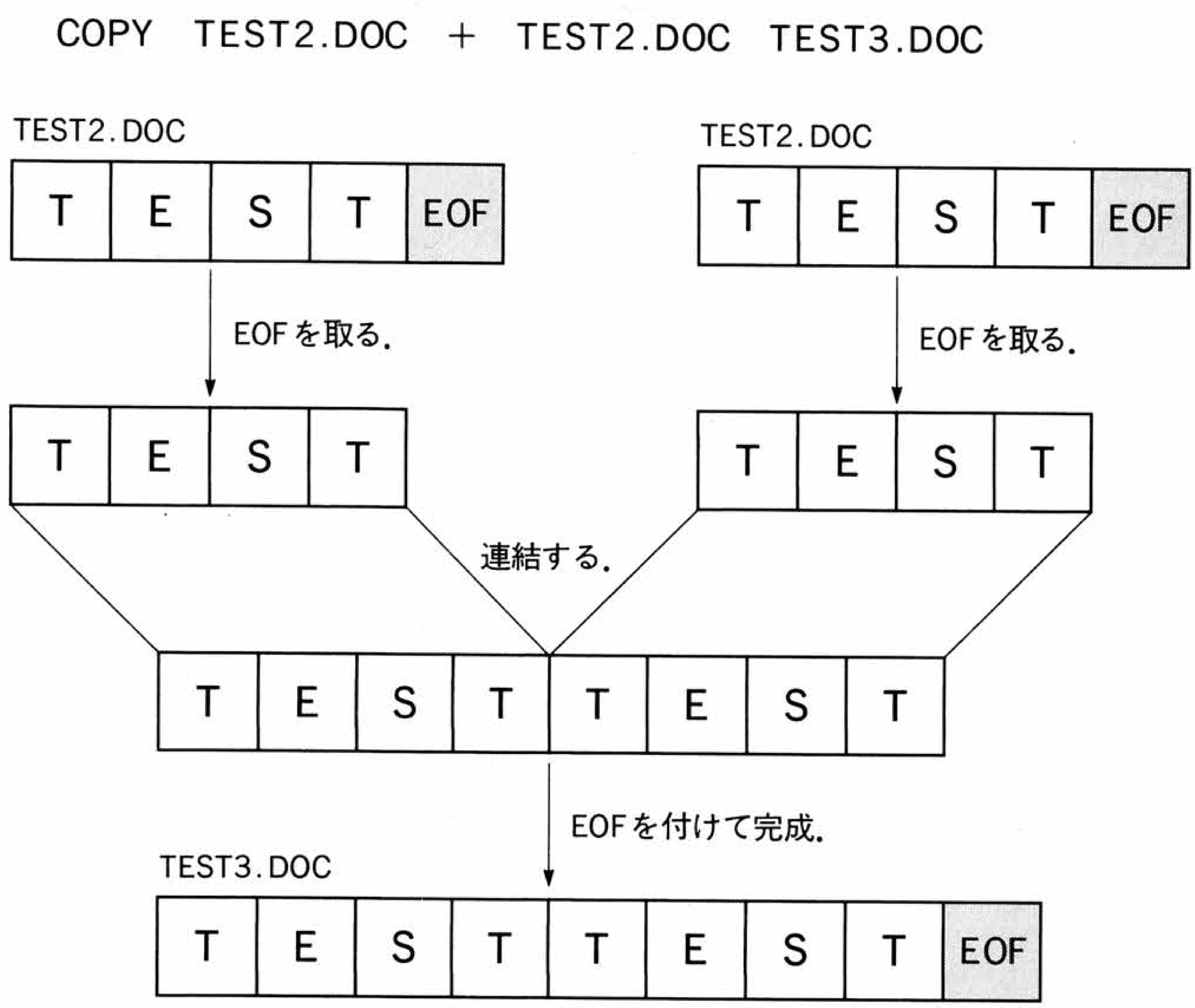


図3.28 アスキー・モードでの連結のしくみ

ファイルの連結が必要となるのは，たいていの場合がテキスト・ファイルなので，とくに何も指定しない場合はすべてテキスト・ファイルだとしてファイルの連結が行われます．しかし，ファイルにはテキスト・ファイル以外にも，COMファイルのような**バイナリ・ファイル**があります．バイナリ・ファイル中にある1AhにはEOFの意味はなく，単にプログラムの一部です．ファイルを連結する際にEOFと勘違いされて，勝手に付けたり取ったりされては困ります．そのため，COPYコマンドには／Aと／Bの2つのスイッチがあります．

〈ファイル・スペック〉の後ろに／Bを付けるとそのファイルはバイナリ・ファイルだとみなされてMSX-DOSはファイルの連結の際にEOFを勝手に付けたり取ったりしません．このように，ファイルに手を加えずに単に連結するモードを**バイナリ・モード**といいます．

また，このスイッチはCOPYコマンドの中で次にスイッチが指定されるまで有効です．途中で／Aスイッチを付けると，その時点でアスキー・モードに戻ります．

バイナリ・モードでのファイルの連結を実際に試してみましょう．**図3.29**の操作を行ってください．

```
A>copy test2.doc /b + test2.doc test4.doc ⏎
        1 file copied
A>type test4.doc ⏎
TEST ……………………………… 前半部分しか表示されない．
A>dir test*.doc ⏎
TEST1    DOC        4 87-12-24  3:48p
TEST2    DOC       [5] 87-12-24  3:48p
TEST3    DOC        9 87-12-24  4:05p
TEST4    DOC      [10] 87-12-24  4:07p ……………… ファイルのサイズは2倍になっている．
        4 files   674816 bytes free
```

図3.29　バイナリ・モードでの連結の実行例

できあがったTEST4.DOCは，TEST2.DOCが2つ連結されたものです．そのため，ファイルのサイズも2倍になっています．しかし，TYPEコマンドで内容を表示すると前半部分しか表示されません．これは，バイナリ・ファイルとみなされたために，**図3.30**のように，最初のTEST2.DOCのEOFもそのままコピーされ，TYPEコマンドでは，そこで表示が打ち切られているためです．

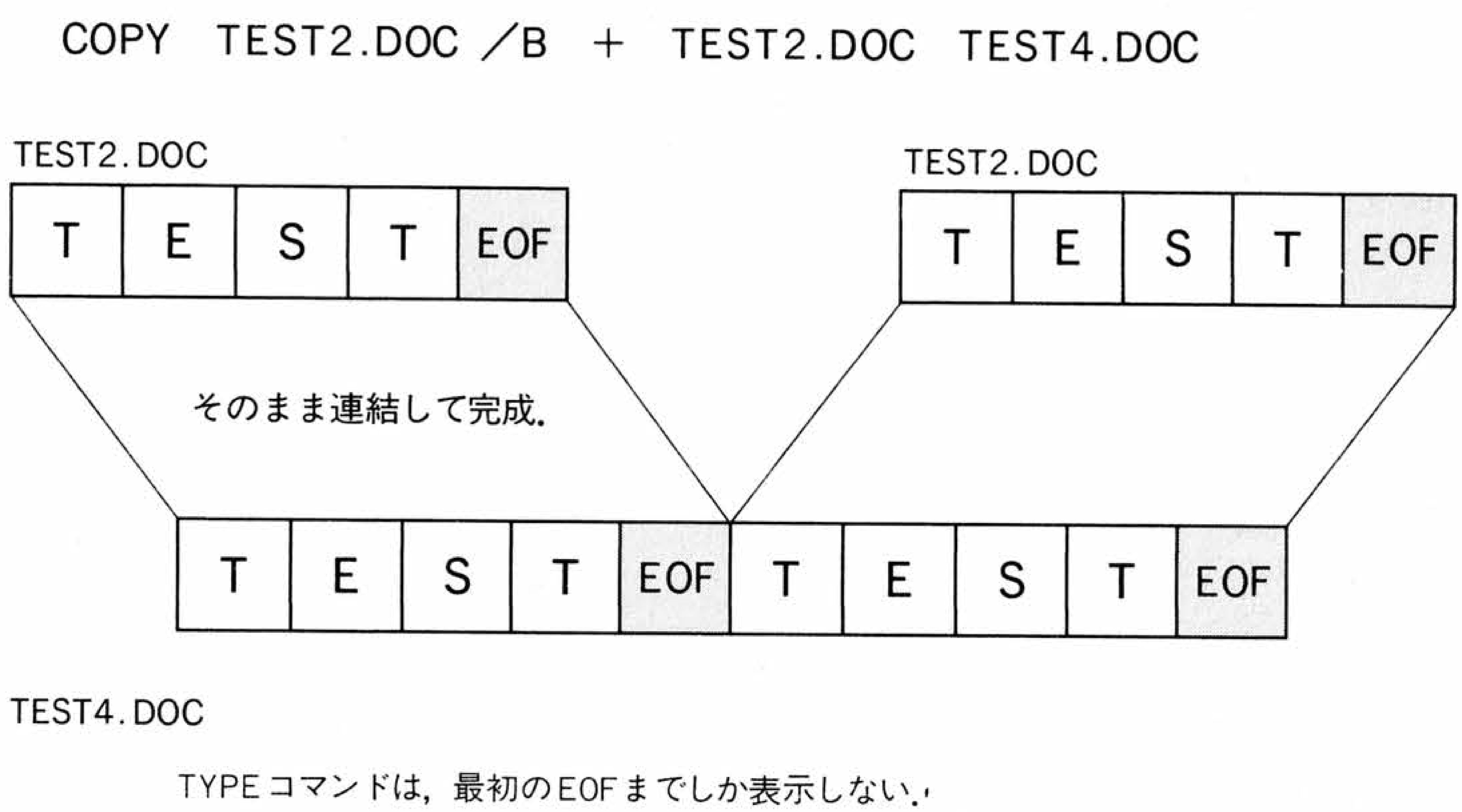


図3.30 バイナリ・モードでの連結のしくみ

ファイルをバイナリ表示で見る

TYPEコマンドではファイルをバイナリとして見ることができず多少不便ですので，ファイルの内容を見るプログラムEDUMPをBASICで書いてみましょう．BASICを起動して，**リスト3.3**のプログラムを入力し，EDUMP.BASというファイル名でセーブします．「MSX-DOS TOOLS」を持っている人はTOOLSのDUMPコマンドを使ってもけっこうです．

```
10 'easy dump '87.9.30        by nue
20 INPUT "file name";FI$
30 OPEN FI$ AS #1 LEN = 1
40 FIELD #1, 1 AS A$
50 CO=LOF(1) : ADDRESS = 0
60 PRINT RIGHT$("0000" + HEX$(ADDRESS), 4);"    ";
70 FOR I = 1 TO 8
80  GET #1
90  PRINT RIGHT$("00" + HEX$(ASC(A$)), 2);" ";
100  CO = CO - 1 : ADDRESS = ADDRESS + 1
110   IF CO = 0 THEN CLOSE : END
120 NEXT
130 PRINT
140 GOTO 60
```

リスト 3.3　EDUMP.BAS

では，EDUMP を使って TEST4.DOC を見てみましょう．BASIC を起動し，〝RUN　"EDUMP.BAS"〟と入力します．すると，内容を表示させたいファイル名を聞いてくるので，**図 3.31** のように入力します．結果を見ると EOF(1Ah)がファイルのまん中にあるのがわかります．

```
file name? test4.doc ⏎
0000     54 45 53 54 [1A] 54 45 53
0008     54 1A          ↑
                        └──EOF
Ok
```

図3.31　EDUMP の実行例

また，TEST1.DOC は最初から EOF のないバイナリ・ファイルですから，バイナリ・モードで連結することが可能です．DOS に戻り，**図 3.32** のように入力してください．

```
A>copy test1.doc /b + test1.doc test5.doc ⏎
     1 file copied  ←バイナリ・モードでの連結を指定.
A>type test5.doc ⏎
TESTTEST
```

図3.32 バイナリ・ファイルをバイナリ・モードで連結

連結されたTEST5. DOCをEDUMPを使って見てみましょう(図3.33).

```
file name? test5.doc ⏎
0000    54 45 53 54 54 45 53 54 ……………EOF(1Ah)がない.
Ok
```

図3.33 バイナリ・モードで連結されたファイルの内容

連結し終わったTEST5.DOCもEOF(1Ah)を持たないバイナリ・ファイルであることがわかります. では, TEST1.DOCを2つ連結してEOFを持つテキスト・ファイルを作ってみましょう(図3.34).

```
A>copy test1.doc /b + test1.doc test6.doc /a ⏎
       1 file copied                      ↑
A>type test6.doc ⏎                        アスキー・モードでの
TESTTEST ……………TYPEコマンドで表示させると,    連結を指定.
                TEST5.DOCと同じに見える.
```

図3.34 バイナリ・ファイルを連結しアスキー・ファイルを作成

この場合, 連結するまではバイナリ・モードですが, デスティネーションのファイル名に／Aスイッチが付いているので, アスキー・モードに戻ります. 結局, TEST6.DOCはEOFの付いたテキスト・ファイルになります(図3.35).

```
file name? test6.doc ⏎
0000      54 45 53 54 54 45 53 54
0008      1A
0k        ↑―――――最後にEOF(1Ah)が付いている.
```

図3.35　EDUMPでEOFを確認

また，ファイルの連結だけでなく，単なるコピーにもこれらのスイッチは有効です．ただし，ファイルのコピーの際には連結の場合とは違って何も指定しないとバイナリ・モードになります(ファイルをコピーする際に勝手に1AhをEOFとみなして，コピーを打ち切られては大変です)．2章でMSXDOS.SYSやCOMMAND.COMをコピーする際に，とくにスイッチを付けなくてよかったのはこのためです．ですから，単なるコピーをアスキー・モードで行いたい場合には，／Aスイッチを付ける必要があります．

また，テキスト・ファイルであっても，ファイルをそのままコピーする場合には普通バイナリ・モードでコピーします(なんのスイッチも付けずにCOPYコマンドを使う)．単にコピーを行うのにEOFを一度取ってから再び付ける必要はありません．そのまま，EOFまでまるごとコピーしてしまうほうが簡単です．

ですから，わざわざ／Aスイッチを付けてコピーを行うのは非常に特殊な場合に限られます．たとえば，**図3.36**のように，EOFのない不完全なテキスト・ファイルからEOFを持つ正常なテキスト・ファイルを作る場合などです．

```
A>copy test1.doc test7.doc /a ⏎
        1 file copied
```

図3.36　バイナリ・ファイルをアスキー・ファイルにコピー

図3.37のように指定するとバイナリ・ファイルであるTEST1.DOCをTEST7.DOCにコピーする際にEOFを付けるので，TEST7.DOCはテキス

ト・ファイルになります(TEST2.DOCの中身と同じになります．DUMPを使ってTEST2.DOCと比べてください)．

また，コピーのモードはファイル・スペックのまえに指定することもできます．／Aを付けて，アスキー・モードでコピーを行うとEOFまでしかコピーされません(図3.37，図3.38)．

```
A>copy /a test4.doc test8.doc ⏎ ……………コピーのモード・スイッチは
       1 file copied                    ファイル・スペックのまえに
A>                                      付けることもできる.
```

図3.37　アスキー・モードでのファイルのコピー

```
file name? test4.doc ⏎
0000     54 45 53 54 1A 54 45 53 ……………コピー元
0008     54 1A
Ok
run ⏎ ……………もう1度EDUMPを起動する.
file name? test8.doc ⏎ ……………コピー先
0000     54 45 53 54 1A ……………EOFまでしかコピーされないことがわかる.
Ok
```

図3.38　アスキー・モードでコピーしたファイルの内容

COPYコマンドの機能についていろいろなことを説明しました．バイナリ・モードとアスキー・モードの違いは，なかなかわかりにくいかも知れません．

しかし，実際にファイルのコピーを行う際には，アスキー・モードを使うことはまずありません．逆に連結を行うのはテキスト・ファイルの場合がほとんどで，バイナリ・モードでの連結を使うことはあまりありません．

そのため，あまりCOPYコマンドのスイッチを使う機会はないと思います．あまり難しく考えずに「必要になったらこの節を読み返せばよい！」ぐらいの気持ちでいればよいでしょう．

3 6 ファイルの削除

ファイルの編集に関するコマンドとしては最後のものとなりましたが，ファイルを削除するコマンドについて説明しましょう．ファイルを削除するには **DEL コマンド**を使います．書式は次のようになります．また **ERASE コマンド**も，DEL コマンドとまったく同じように使えます．

書式　DEL　〈ファイル・スペック〉
書式　ERASE　〈ファイル・スペック〉

実際にファイルを削除してみましょう．**図 3.39** のように入力します．

```
A>dir test1.doc ↵
TEST1     DOC        4 87-12-24   3:48p
        1 file   670720 bytes free
A>del test1.doc ↵
A>dir test1.doc ↵
File not found ……………ファイルが削除されたので見つからない．
A>
```

図 3.39　DEL コマンドの実行例

図 3.40 のように，TEST1.DOC は消えてしまいました．また，ワイルドカードも使えます．**図 3.40** のように指定すれば，ファイル名が TEST で始まるファイルのすべてが削除されます．

```
A>dir
SAMPLE   3        6656 85-09-02 10:10p
SAMPLE   2        6656 85-09-02 10:10p
MSXDOS   SYS      2432 85-08-23  9:29p
COMMAND  COM      6656 85-09-02 10:10p
TEST     2        2432 85-08-23  9:29p
SAMPLE   ONE      2432 85-08-23  9:29p
MSXDOS   BAK      2432 85-08-23  9:29p
MSX!DOS  SYS      2432 85-08-23  9:29p
MSXCOM   ONE      6656 85-09-02 10:10p
COMMAND  BAK      6656 85-09-02 10:10p
TEST2    DOC         5 87-12-24  3:48p
TEST3    DOC         9 87-12-24  4:05p
TEST4    DOC        10 87-12-24  4:07p
TEST5    DOC         9 87-12-24  5:40p
TEST6    DOC         9 87-12-24  5:48p
TEST7    DOC         5 87-12-24  5:49p
TEST8    DOC         5 87-12-24  5:56p
      17 files   671744 bytes free
A>del test*.*
A>dir
SAMPLE   3        6656 85-09-02 10:10p
SAMPLE   2        6656 85-09-02 10:10p
MSXDOS   SYS      2432 85-08-23  9:29p
COMMAND  COM      6656 85-09-02 10:10p
SAMPLE   ONE      2432 85-08-23  9:29p
MSXDOS   BAK      2432 85-08-23  9:29p
MSX!DOS  SYS      2432 85-08-23  9:29p
MSXCOM   ONE      6656 85-09-02 10:10p
COMMAND  BAK      6656 85-09-02 10:10p
       9 files   681984 bytes free
A>
```

TEST＊.＊にマッチするファイル

指定したファイルが消えている.

図3.40　ワイルドカードを使ったファイルの削除

また，**図3.41**のようにするとディスク上のすべてのファイルを削除してしまいます.

```
A>del *.*
Are you sure (Y/N)? n
A>dir
SAMPLE   3      6656 85-09-02 10:10p ┐
SAMPLE   2      6656 85-09-02 10:10p │
MSXDOS   SYS    2432 85-08-23  9:29p │
COMMAND  COM    6656 85-09-02 10:10p │
SAMPLE   ONE    2432 85-08-23  9:29p ├ 消えていない.
MSXDOS   BAK    2432 85-08-23  9:29p │
MSX!DOS  SYS    2432 85-08-23  9:29p │
MSXCOM   ONE    6656 85-09-02 10:10p │
COMMAND  BAK    6656 85-09-02 10:10p ┘
       9 files   681984 bytes free
A>del *.*
Are you sure (Y/N)? y
A>dir
File not found ・・・・・・・・・・・・・・消えている.
A>
```

図3.41　すべてのファイルを削除

図3.41のように，すべてのファイルを削除するときだけは，確認を求めてきます．ここで，[N]と入力すればファイルの削除を行わずにコマンド・レベルに戻りますが，[Y]と入力すると本当にすべてのファイルが消えてしまいます．

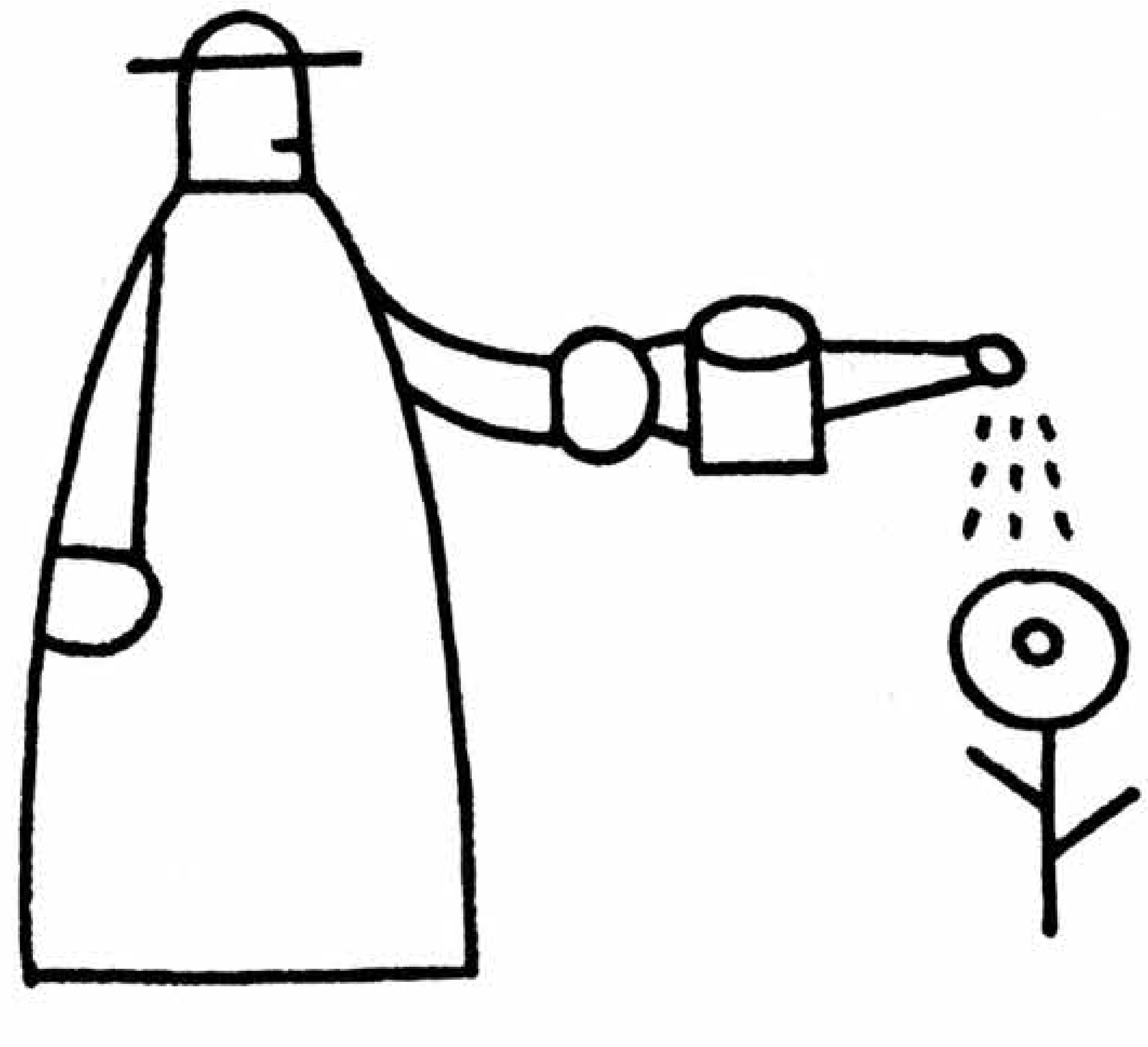
ファイルを削除するコマンドはなくてはならないものですが不用意に使うとこれ以上危険なコマンドはありません．使う際には本当に削除してよいファイルなのかを確認してから使うように心がけてください．

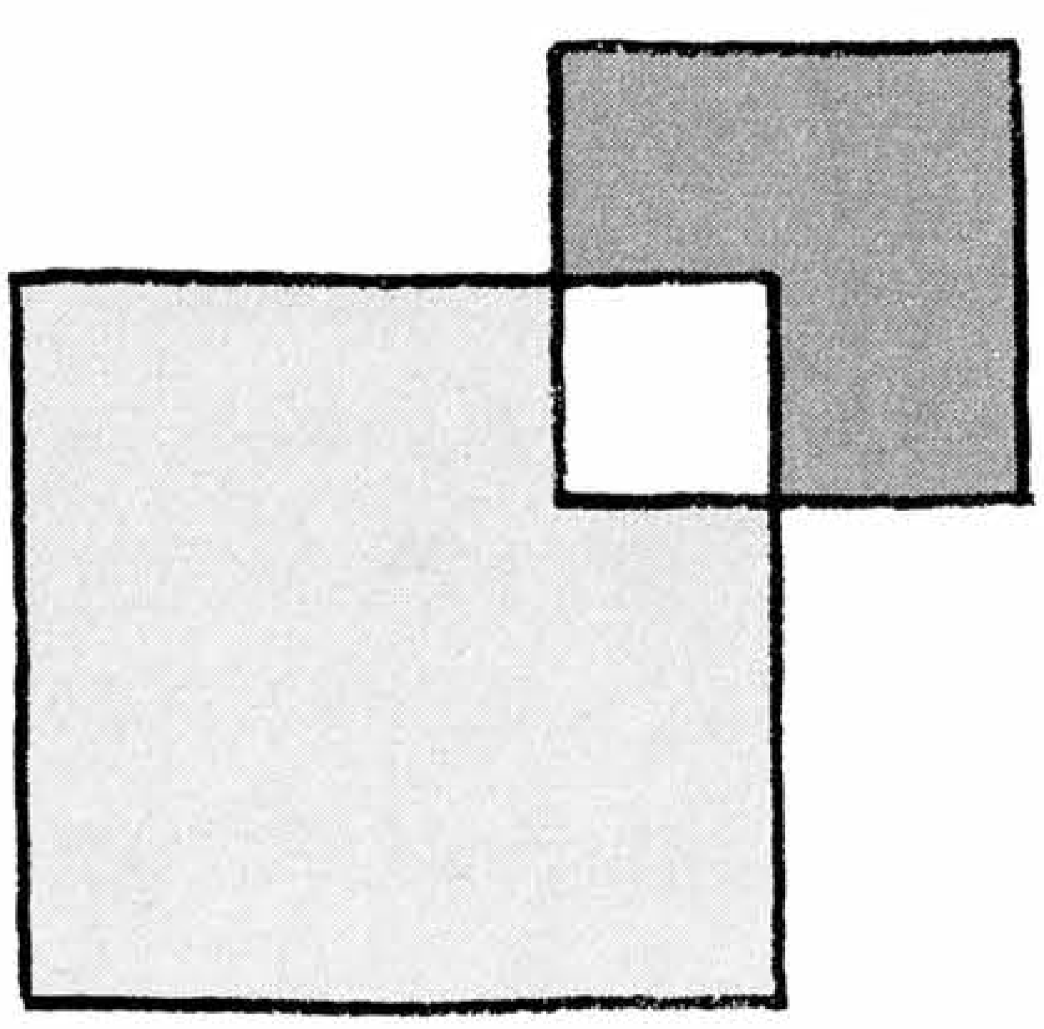
以上でディスク上のファイル管理に関するコマンドの説明はだいたい終わりました．次章では，ディスク上のデータやプログラム以外のファイルについて見ていくことにしましょう．

4章

周辺装置との データのやりとり

―デバイス・ファイルを使う―





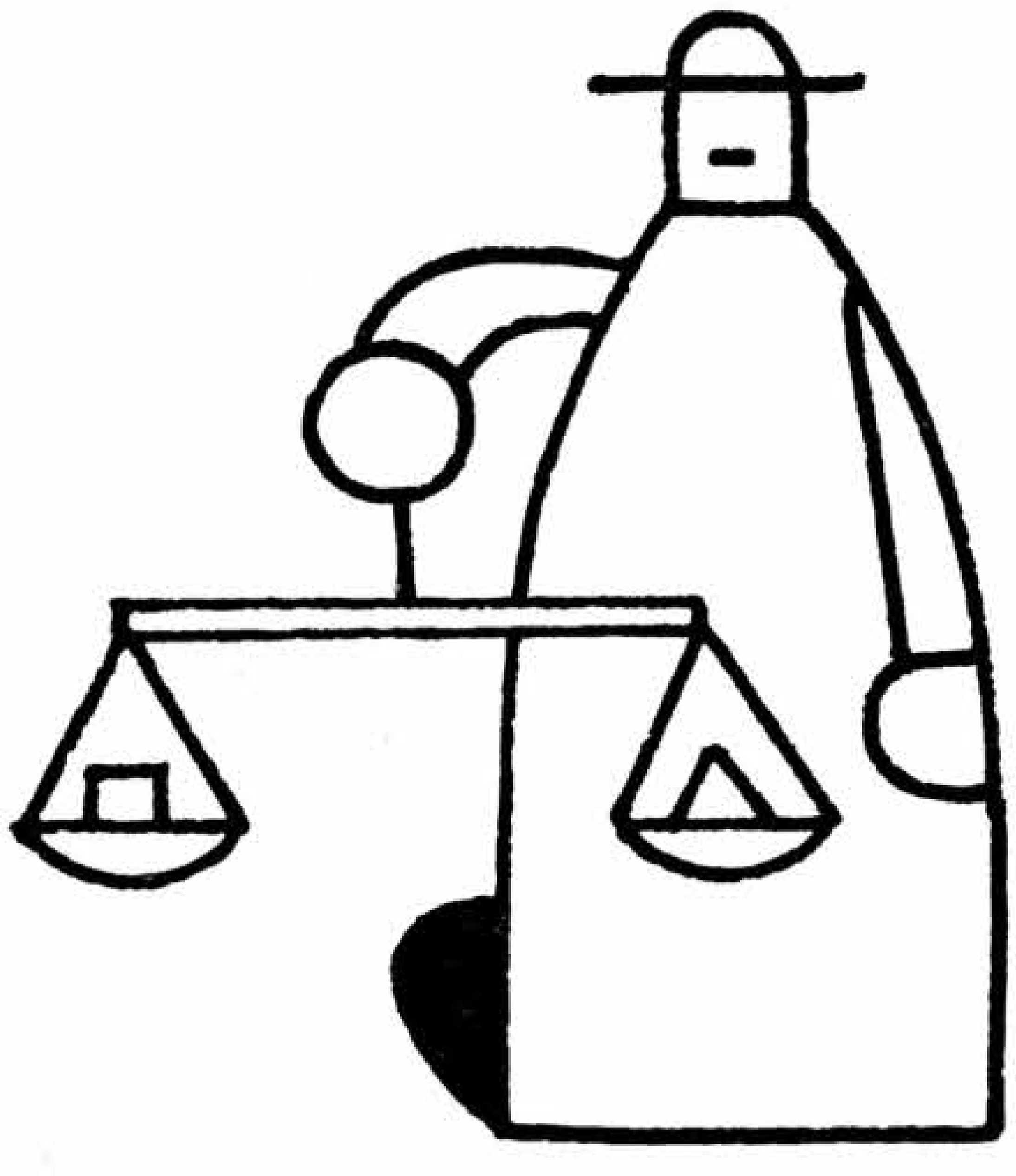
4 1 デバイス・ファイル

ディスク上のファイルについて3章までで説明してきましたが，MSX-DOSで扱うファイルはディスク上のファイルだけではありません．この章では，それ以外のファイル，つまり周辺装置を表すデバイス・ファイルというものについて説明していくことにしましょう．

3章で出てきたMSX-DOSのさまざまなコマンドの中でも，最も重要なのはCOPYコマンドです．このコマンドはデータをディスク上のあるファイルから別のファイルへと自由にコピーできるというものでした．

つまり，COPYコマンドを使うことにより，ディスク・ドライブ装置から読み出したデータをディスク・ドライブ装置に送り込むことができる，と考えることができます．

ところで，MSXがデータのやりとりをする周辺装置は，何もディスク・ドライブだけではありません．キーボードやディスプレイ，プリンタなどともデータのやりとりを行います．



DOSから見れば，これらの周辺装置とディスク・ドライブを区別する必要性はありません．周辺装置が変わっても，データを読み込んだり，送り出したりしていることに変わりはありません．キーボードから読み込んだデータも，ディスクから読み込んだデータも，まったく違いはないのです．それならば，キーボードからデータを読み込んでディスク上にファイルを作る命令も，ディスクからデータを読み込んでファイルを作る命令も区別せず，同じCOPYコマンドを使えるようにしてもよいのではないでしょうか．

このような考え方から生まれたのがデバイス・ファイルです．キーボードやプリンタなどの周辺装置もファイルの1種と考えることにします．こうすれば，COPYコマンドがそのまま使えるので，周辺機器にとらわれることなく，データを扱うことが可能になります．**図4.1**を見てください．

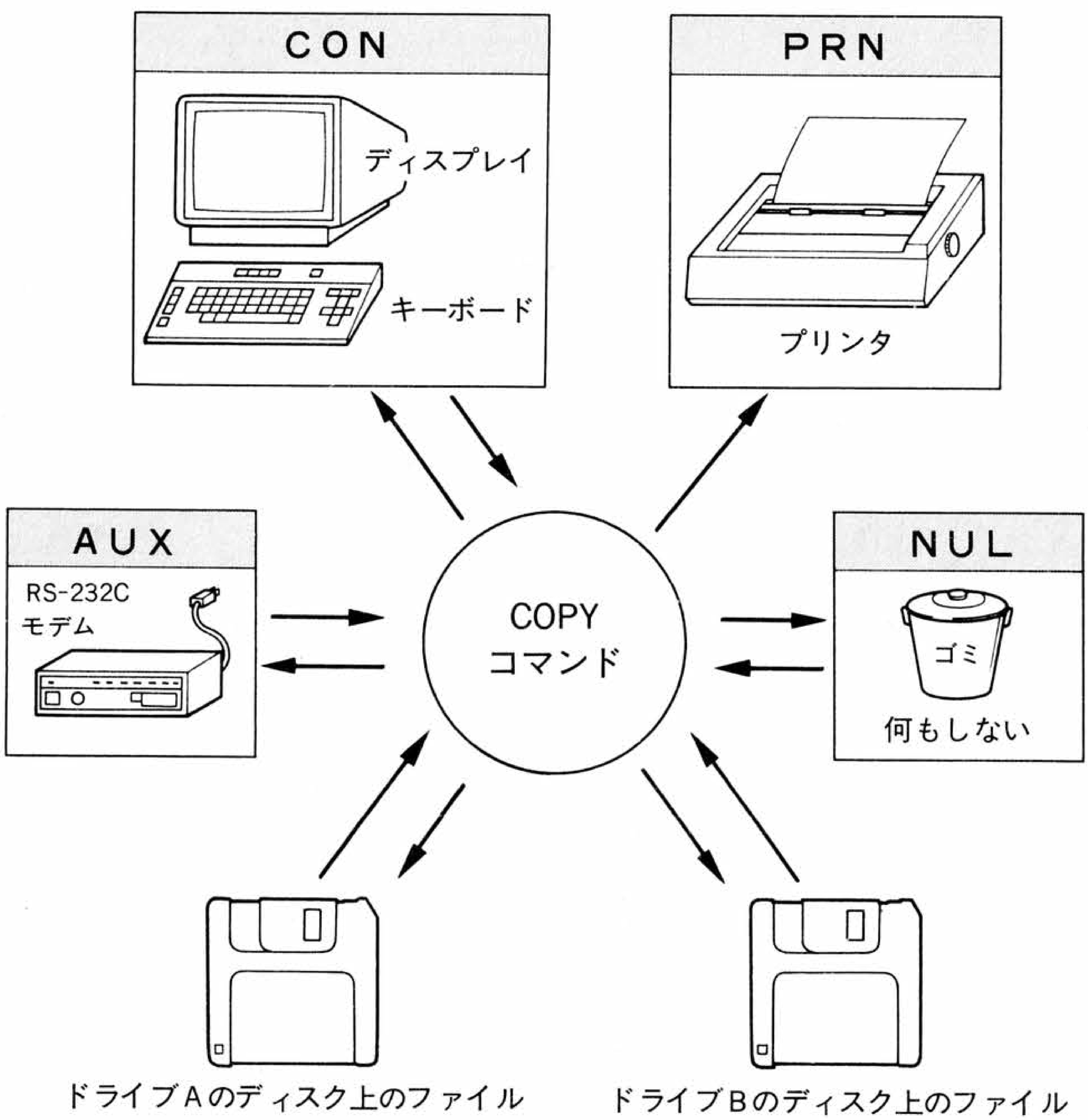


図4.1　デバイス・ファイル概念図

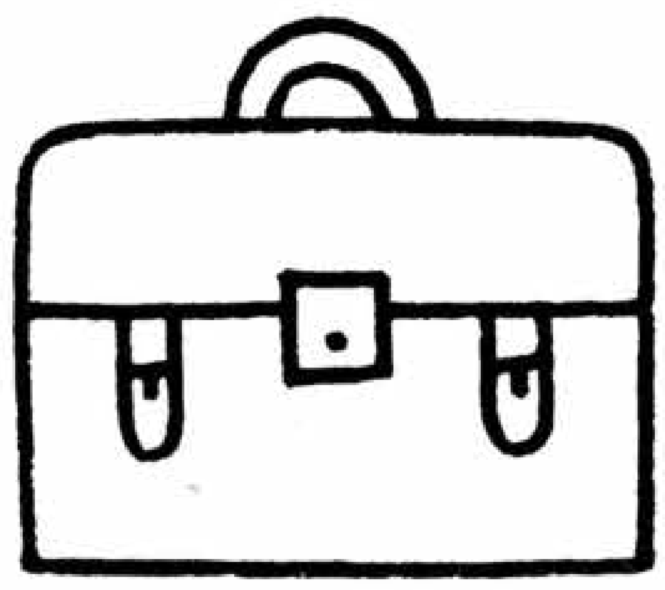
MSX-DOSでは4種のデバイス・ファイルがあります．これらは**表4.1**のように，それぞれの周辺機器と対応しています．

ファイル名	データの向き	働　　き
CON	入　力	キーボードからの入力
	出　力	ディスプレイへの出力
PRN	出　力	プリンタへの出力
NUL	入出力	何もしない
AUX	入出力	補助入出力装置との入出力

表4.1　デバイス・ファイル表

これらのデバイス・ファイルはディスク上にあるものではなく，MSX-DOSが立ち上がったときから自動的に用意されているものです．また，CONはキーボードとディスプレイの両方と対応していますが，これはデータを入力するときと出力するときで対応する周辺装置が異なるためです．詳しくは次節で説明します．

では，最も基本的なCONから見ていくことにしましょう．



4 2 コンソールとの入出力

コンソール(キーボードとディスプレイの両方を合わせて指し示す名前)と対応するデバイス・ファイルは**CON**です．

CONは入力(コピー元：ソース)のときにはキーボードを表し，出力(コピー先：デスティネーション)のときにはディスプレイを指します．

これは，キーボードにデータを送ったり，ディスプレイからデータを読み込んだりする必要が生じることはあり得ませんから，容易に納得することができます．

では，実際にCONを使ってみましょう．CONからファイルをコピーすれば，キーボードからデータを読み込んでくれるので，簡単にファイルを作ることができます．

図4.2では，コンソール(キーボード)からデータを読み込みその結果をCONSOLE.DOCというディスク上のファイルにしています．

```
A>copy con console.doc ↵
dir ↵
copy msxdos.sys b: ↵
commands are ignored! ↵
nanno!! ↵
```

(dir 〜 nanno!! の行：ファイルCONの内容．)

図4.2　キーボードからディスク上のファイルにコピー

COPYコマンドを実行すると文字を入力することはできてもコマンドを実行することはできない状態になります．ここで入力した文字はコマンドとしてではなくテキスト・ファイルの中身としてMSX-DOSに認識されています．

コマンド・レベルに戻るにはテキスト・ファイルの最後だということをDOSに教えてやらなくてはなりません．つまり，EOFを入力すればよいのです．キーボードからEOFを入力するには［CTRL］＋［Z］を入力します(コントロールコードで，1AhはCTRL-Zに割り当てられている)．

```
A>copy con console.doc
dir
copy msxdos.sys b:
commands are ignored!
nanno!!
^Z  ……………EOF(1Ah)を入力．[CTRL]+[Z]を押す．
        1 file copied
A>dir ⏎
CONSOLE  DOC       58 87-02-12  4:56p
        1 file    730054 bytes free
A>
```

図4.3　EOFの入力

このように，EOFを入力するとファイルをコピーしてコマンド・レベルに戻ります．本当にCONSOLE.DOCというファイルができているかどうか確かめるため，TYPEコマンドを使ってファイルの内容を表示させてみましょう．図4.4のように入力してください．

```
A>type console.doc ⏎
dir
copy msxdos.sys b:
commands are ignored!
nanno!!

A>
```

図4.4　作成したファイルの内容を確認

さきほど入力した結果がファイルの内容になっています．このようにテキスト・ファイルを最も簡単に作るにはCONからのコピーを使えばよいのです．

さて，いま作ったCONSOLE.DOCを，今度はCONに対してコピーしてみましょう．**図4.5**のように入力します．

```
A>copy console.doc con ↵
dir
copy msxdos.sys b:
commands are ignored!
nanno!!
        1 file copied ……………TYPEコマンドと異なり，コピー終了の
                              メッセージが表示される．
A>
```

図4.5 COPYコマンドでディスク上のファイルを画面に表示

今度は即座にコピーが行われ，ファイルの中身がディスプレイに表示されました．このように，CONは出力先に指定されると，ディスプレイに対してデータを送り込みます．このままでは，TYPEコマンドと同じ機能のように見えますが，COPYコマンドにはファイルの連結という機能がありました．これを使えば，**図4.6**のようなことができます．

```
A>copy con con2.doc ↵               ┐
*********************** ↵            │
*                     * ↵            │
*       con2.doc      * ↵            ├ ファイルCON2.DOCを作成．
*                     * ↵            │
*********************** ↵            │
^Z ↵                                 ┘
        1 file copied
A>copy con2.doc+console.doc con ↵ …………CON2.DOCとCONSOLE.DOC
                                     をCONにコピー．
```

```
**********************
*                    *
*      con2.doc      *
*                    *
**********************
dir
copy msxdos.sys b:
commands are ignored!
nanno!!
       1 file copied
A>
```

連結されて画面に表示される.

図 4.6 ファイルを連結して画面に表示

図 4.6 では, CON2.DOC と CONSOLE.DOC を連結し, CON にコピーしています. また, キーボードからの入力と他のファイルの連結も可能です. 図 4.7 では, コンソールからの入力に CONSOLE.DOC を連結して CON3.DOC というファイルにしています.

```
A>copy con+console.doc con3.doc
M S X - D O S !!
^Z
       1 file copied
A>type con3.doc
M S X - D O S !!
dir
copy msxdos.sys b:
commands are ignored!
nanno!!

A>
```

キーボードからの入力とディスク上のファイルが連結されている.

図 4.7 キーボードからの入力とファイルを連結

CON を使えば, このように, テキスト・ファイルのまえや後ろに文字を付け足したり, 複数のテキスト・ファイルをつなげて読むこともできます.

4-3 プリンタへの出力

MSX-DOSでは，プリンタもファイルとして扱われます．デバイス・ファイルPRNがそうです．使い方はPRNもCONもそれほど変わりはないのですが，PRNは出力先としてしか使えません．

プリンタを持っている人は，**図4.8**のコマンドを試してみてください．

```
A>copy console.doc prn ⏎
        1 file copied ……………プリンタに出力されている．
A>
```

図4.8 ディスク上のファイルをプリンタにコピー

プリンタが正しくつながっていれば**図4.9**のように，CONSOLE.DOCの内容がプリンタに打ち出されるはずです．

```
dir
copy msxdos.sys b:
commands are ignored!
nanno!!
```

図4.9 PRNへのコピー結果(プリンタ出力)

プリンタをつながずにこのコマンドを実行するとハング・アップ(MSXが何の反応も示さなくなり，キー入力も受け付けなくなる)してしまいます．この場合はリセット・ボタンを押して，MSX-DOSを再起動させてください(リセット・ボタンのないMSXでは，電源を一度切ってください)．

COPY コマンドの連結機能を使える点は CON と同じです(**図 4.10，図 4.11**).

```
A>copy console.doc + con2.doc prn ⏎
      1 file copied
A>
```

図 4.10　ファイルを連結してプリンタへ出力

```
dir
copy msxdos.sys b:
commands are ignored!
nanno!!
***********************
*                     *
*       con2.doc      *
*                     *
***********************
```

図 4.11　連結され PRN へコピーされたファイル(プリンタ出力)

PRN にファイルをコピーすることは，プリンタにテキスト・ファイルの中身を打ち出させることを意味します.

また，PRN は LST という別名を持っています．どちらを使っても結果は同じですが，MS-DOS との互換性を考えるならば，PRN で慣れておいたほうがよいでしょう.

ここで，残りのデバイス・ファイル **NUL** と **AUX** についても説明してしまいましょう．まずデバイス・ファイル NUL は入出力の両方に使えるデバイス・ファイルなのですが，どちらに指定しても何もしません.

NUL は何もしないというファイルなのです．**図 4.12** を実行してください.

```
A>copy nul nulfile.doc ⏎
        1 file copied
A>type nulfile.doc ⏎
                                  ……………空ファイルができている.
A>dir nulfile.doc ⏎
NULFILE  DOC         0 87-10-18   1:06a
        1 file   699392 bytes free
A>copy console.doc nul ⏎
        1 file copied
A>dir nul*.* ⏎
NULFILE  DOC         0 87-10-18   1:06a ……NULというファイルは
        1 file   699392 bytes free        できていない.
A>
```

図 4.12　NULへのコピーとNULからのコピー

図 4.12 のように，NUL からファイルをコピーしても空のファイルができるだけですし，NUL にファイルをコピーしてもなんの変化もありません．

なんのためにこのようなデバイス・ファイルがあるかというと，コマンドが入出力先にファイル名を要求しているにもかかわらず，とくにファイルを指定したくない場合に使います．

MSX-DOS に最初から組み込まれている内部コマンドだけを使っているぶんには，このようなものは必要ありませんが，6 章で紹介する「MSX-DOS TOOLS」を利用する場合や，7 章で紹介する「MSX-C」を使う場合などに必要になります．

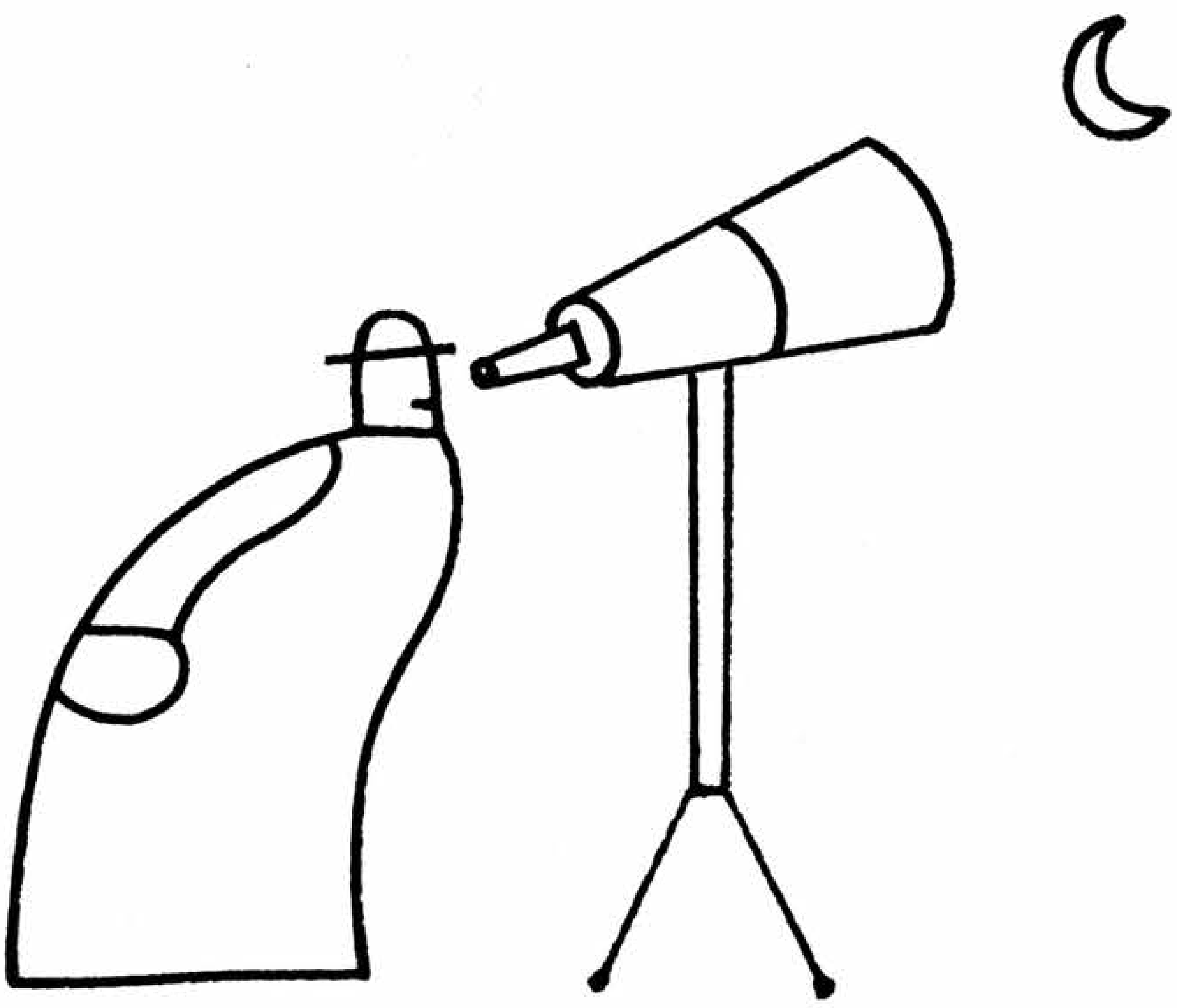
NUL を使う機会はあまりないかもしれませんが，何もしないデバイス・ファイルがあるということは覚えておいてください．

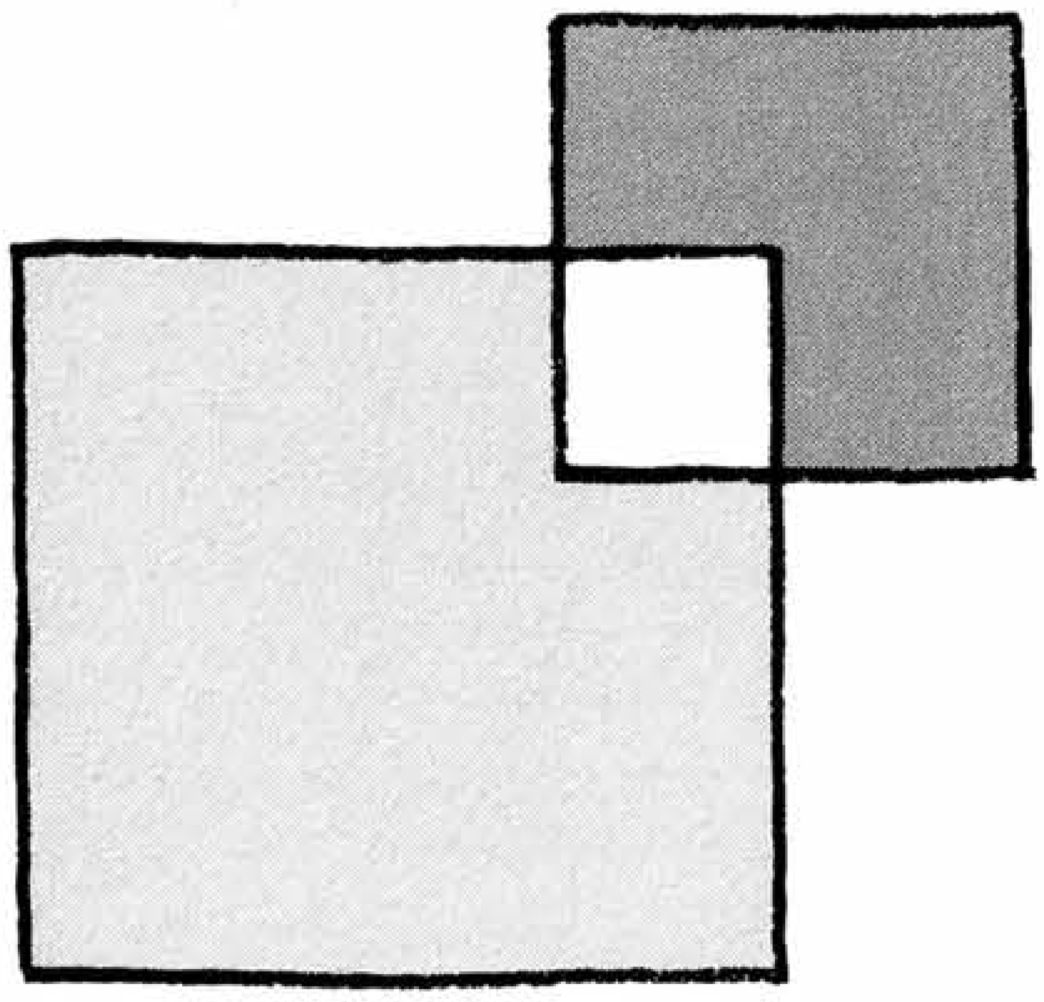
また，AUX は，RS-232C などの外部補助入出力のために用意されているデバイス・ファイルです．しかし，MSX-DOS 起動時には NUL と同じ機能に設定されていますので，何もしないデバイス・ファイルになっています．そのため，この名前も普通のファイル名としては使用することはできません．

5章

バッチ・ファイルを使う

―コマンドの一括処理―





5-1 バッチ・ファイルとは

MSX-DOSの基本的な使い方は前章までで説明し終わりました．本章では少し応用的な機能であるバッチ処理を使ってみることにしましょう．

MSX-DOSを使い込んでくると，毎回同じ手順でコマンドを与えるのがわずらわしく感じられることがあります．たとえば，DOSを起動させる際にディスプレイの表示文字幅を設定したり，日付，時刻の確認を行いますが，そのたびにいくつものコマンドを入力するのは，あまり能率的とはいえません．また，システム・ディスクのバックアップなどは，DOSを使っている限り，頻繁に行わなければならない作業の1つです．しかし，実際にその作業を追ってみると，ある決まったコマンドを，毎回同じ手順で実行しなければなりません．いくつかの決まったコマンドを順番に実行していく処理も，簡単なものであればその場で実行すればよいのですが，処理が複雑になり，コマンド数が増えてくると，一連のコマンドをいちいち入力していくのがめんどうになってきます．また，複雑なコマンド操作になると，書式や順番を間違える可能性もあります．

このわずらわしい作業から，われわれを解放してくれるのが**バッチ処理**です．バッチ処理とは，一般に，あらかじめ蓄えられた処理を一括して行うことを指します．

MSX-DOSでは，コマンドを作業の手順に従って**バッチ・ファイル**と呼ばれるファイルに登録しておくことにより，そのファイル名を指定するだけで一連の処理を実行してくれます．このバッチ処理機能を使えば複数のコマンドを組み合わせて，複雑な処理をさせる1つのコマンドとして使うことも可能です．これは，別にマシン語の知識がなくても，既存のコマンドの組み合わせにより，新たなコマンドを簡単に増やしていくことができるので，とても便利です．また，いろいろなバッチ・ファイルを用意することにより，MSX-DOSでの作業能率が飛躍的に向上します．

具体的には，実行させたいコマンドを書き並べたテキスト・ファイルを作ります．このファイルのメイン・ファイル名は，ファイル名に許された範囲でどのような名前を付けてもよいのですが，拡張子はBATである必要があります．

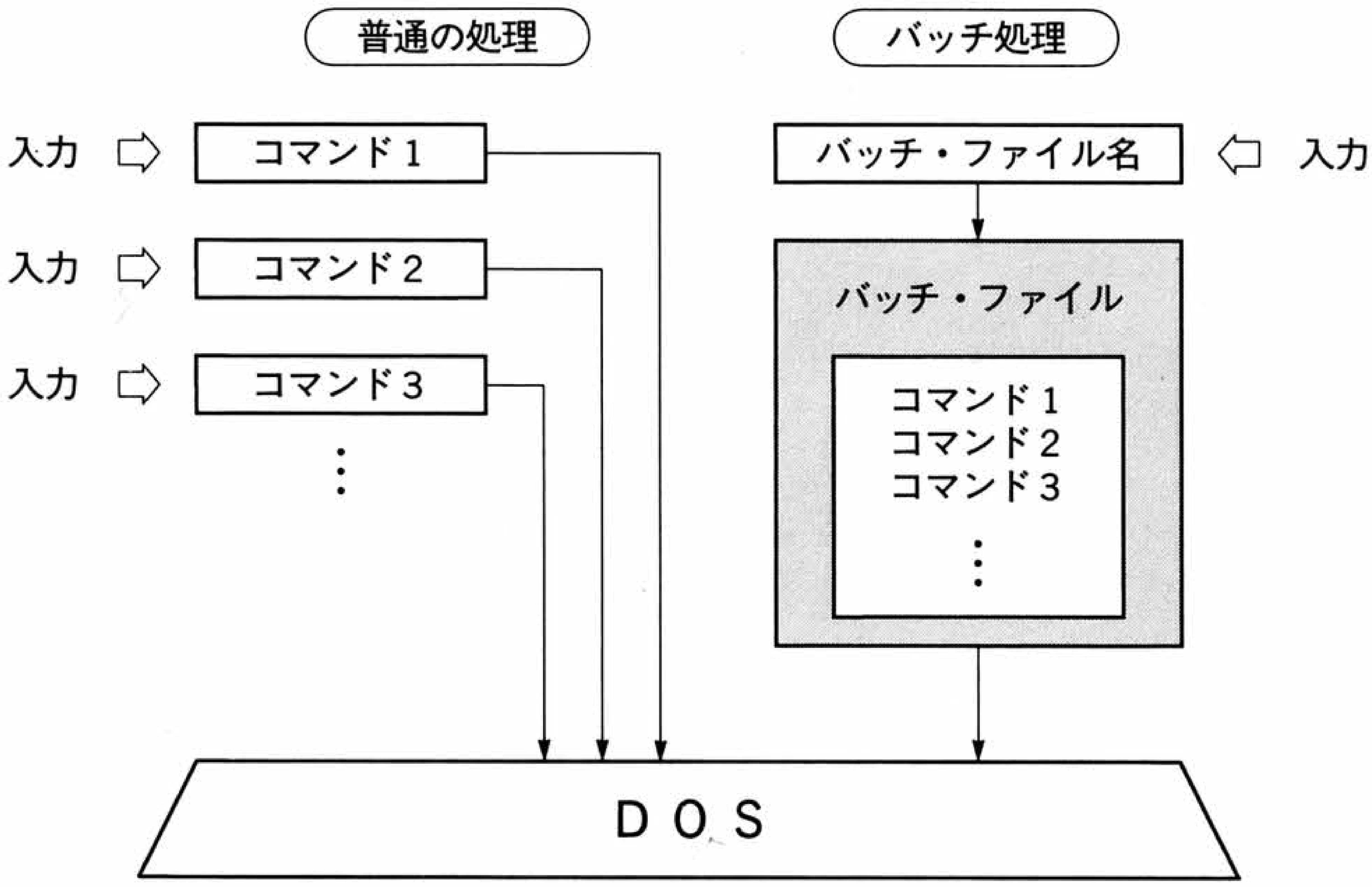


図5.1　バッチ・ファイル概念図

このバッチ・ファイルができたら，あとはCOMファイルやコマンドを実行させるのと同じように，メイン・ファイル名をコマンド・レベルで入力するだけでバッチ処理が実行されます．

では，実際にバッチ・ファイルを作ってみましょう．

5-2 バッチ・ファイルの作成

とりあえず，簡単な例としてシステム・ディスクを作るという処理内容のバッチ・ファイルを作ってみましょう．2章で作ったシステム・ディスクのバックアップ・ディスクをドライブAに入れ，それとは別に新しいディスク(3章でCOPYの練習に使ったディスクでもけっこうです)を用意してください．

システム・ディスクを作るバッチ・ファイル

バッチ・ファイルを作るのは普通のテキスト・ファイルを作るのとまったく同じです．エディタやBASICを使ってもよいのですが，ここではデバイス・ファイルCONを使いましょう．**図5.2**のように入力します．

```
A>copy con msys.bat ⏎
pause 'INSERT NEW DISK INTO B-DRIVE' ⏎ ………メッセージを表示して
                                               一時停止する．
format ⏎ ……………………………フォーマットさせる．
copy a:msxdos.sys b: ⏎  } DOSのシステムをコピーする．
copy a:command.com b: ⏎ }
^Z ⏎
        1 file copied
A>
```

図5.2　バッチ・ファイルの作成

こうしてMSYS.BATができました．ここで初めて出てきたのが**PAUSEコマンド**です．このコマンドが実行されるとメッセージを表示してバッチ処理を一時停止します．

書式は次のようになります．

書式　PAUSE　〈メッセージ〉

〈メッセージ〉はとくになくてもかまいません．また，似たようなコマンドにコメントを表示するだけで一時停止はしない**REMコマンド**もあります．

書式　REM　〈コメント〉

では，システム・ディスクを間違ってフォーマットしてしまわないようにライト・プロテクト・ノッチを書き込み禁止にしてください(フォーマットするディスクは書き込みを禁止しないでください)．MSYS.BATを実行してみます．**図5.3**のようにメイン・ファイル名を入力してください，

```
A>msys ⏎ ……………MSYS.BATの起動．
A>pause 'INSERT NEW DISK INTO B-DRIVE'
Strike a key when ready ……………PAUSEコマンドで停止している．
```

図5.3　MSYS.BATの実行例

図5.3のように，PAUSEコマンドのところでバッチ処理は一時停止します．ここで何か任意のキーを押せば実行を再開します．もし，バッチ処理を中止したい場合は CTRL + C を押します．別にPAUSEコマンドの実行中以外でも CTRL + C は効きますが，バッチ・ファイルを止める可能性のあるところではPAUSEを挟んでおくほうがよいと思います．さて，ここで CTRL + C を押すと，**図5.4**のようなメッセージが表示され本当にバッチ処理を中止するのか確認を求めてきます．

```
A>msys
A>pause 'INSERT NEW DISK INTO B-DRIVE'
Strike a key when ready ^C ……………CTRL+C
Terminate batch file (Y/N)? y …………バッチ処理を中断するかどうか聞い
                                      てくるので Y を入力.
A> ……………コマンド・レベルに戻る.
```

図 5.4 バッチ処理の中止

ここで，Y を入力すると本当にバッチ処理は中止され，コマンド・レベルに戻ります．N を押すとバッチ処理の実行を続けます．

ここでは，Y を押したのでコマンド・レベルに戻ってしまいました．もう一度 MSYS.BAT を実行させます．

今度は PAUSE のメッセージに従って新しいディスクをドライブ B にセットしてください．

何か適当なキーを押すと FORMAT コマンドが実行されます．ドライブ B のディスクを 2 章で行ったのと同じ要領でフォーマットしてください(図 5.5)．

```
A>msys ↵
A>pause 'INSERT NEW DISK INTO B-DRIVE'
Strike a key when ready ……………このメッセージが表示されたら,
A>format                          任意のキーを押す
Drive name? (A,B) b

1 - 1 side
2 - 2 sides
3 - 1 side, double track
4 - 2 sides,double track

? 4
Strike a key when ready
Format complete
```

```
A>copy a:msxdos.sys b:
        1 file copied
A>copy a:command.com b:
        1 file copied
A>
```

図 5.5　MSYS.BAT による FORMAT コマンドの起動

図 5.5 のように，フォーマットが終了すると自動的にシステムがコピーされます．仮想ドライブ機能を使っている人はメッセージに従ってディスクを入れ替えてください．

システムのコピーが終わると，バッチ処理をすべて終了するのでコマンド・レベルに戻ってきます．

このように，MSYS.BAT を使えばシステム・ディスクのバックアップ・ディスクを簡単に作ることができます．では，いま MSYS.BAT を使って新しく作ったシステム・ディスクをドライブ A に入れてください．

何か別のバッチ・ファイルを作ってみましょう．バッチ・ファイルには引数を渡すこともできるので，この機能を使ってみることにします．

では，ある拡張子を持つファイルのディレクトリを表示して，そのファイルの拡張子を何か別のものに変えるバッチ・ファイル EXCHANGE.BAT を作ってみます．**図 5.6** のように入力します．

```
A>copy con exchange.bat ⏎
dir *.%1 ⏎
ren *.%1 *.%2 ⏎
dir ⏎  ↑       ↑───────仮引数
^Z ⏎
        1 file copied
A>
```

図 5.6　拡張子を変更するバッチファイル

入力できたら，実行してみましょう．**図 5.7** のように入力してください．

```
A>exchange sys bak ⏎
A>dir *.sys
MSXDOS   SYS     2432 85-08-23  9:29p
       1 file   718848 bytes free
A>ren *.sys *.bak
A>dir
MSXDOS   BAK     2432 85-08-23  9:29p ……………拡張子が変更された.
COMMAND  COM     6656 85-09-02 10:10p
EXCHANGE BAT       31 87-12-24  4:58p
       3 files   718848 bytes free
A>
```

図 5.7　EXCHANGE.BAT の実行例

バッチ・ファイル中に登場する〝%1〟とか〝%2〟というのが，**仮引数**と呼ばれるものです．仮引数は%0～%9 の 10 個が使えます．このうち，%0 はバッチ・ファイル自身のファイル名がはいり，%1 から%9 はバッチ・ファイルを呼び出す際に指定された実引数と順番に置き替えられます．つまり，バッチ・ファイルは引数を 9 個までとることができるのです．

図 5.7 の例では，実際にバッチ・ファイルが実行されるときには%1 は SYS, %2 は BAK と置き替えられているわけです．

では，このバッチ・ファイルを使って BAK に変えてしまった拡張子を SYS に戻してみてください．やり方はもうわかりますね．〝EXCHANGE BAK　SYS〟を実行すればよいのです．

主にバッチ・ファイルで使うコマンド

ここでいままでに説明しなかったコマンドについて，まとめて説明してしまいましょう．これらは，コマンド・ラインから直接実行されることもありますが，主にバッチ・ファイルの中で使われるものです．

● MODE コマンド

書式　MODE　〈表示文字幅〉

ディスプレイの画面表示幅を指定するコマンドです．〈表示文字幅〉には，MSX なら 1～40 まで，MSX2 なら 1～80 までの整数が指定できます．

設定した，表示幅が 32 文字以下ならスクリーン・モード 1，33 文字以上ならスクリーン・モード 0 となります．また，このコマンドを実行すると画面はクリアされます(図 5.8)．

```
A>mode 32 ↵          このあと画面はクリアされてモードが変わる.
```

図 5.8　MODE コマンド

● DATE コマンド

書式　DATE　〈年〉－〈月〉－〈日〉

MSX-DOS が管理する内蔵カレンダ(内蔵カレンダを持っていない MSX でも設定はできます)の日付を設定もしくは表示します．引数として〈年－月－日〉を指定するとその日付にカレンダをセットしコマンド・レベルに戻りますが，もし引数を省略すると現在の曜日，日付を表示してから新しい日付の入力を求めてきます(図 5.9)．

図 5.9 のように曜日と西暦の下 2 桁，月，日が表示されます．日付を変えたくなければ単にリターンを入力してやれば，コマンド・レベルに戻ります．

引数として日付を指定するとき，〈年〉は 1980～2079，〈月〉は 1～12，〈日〉は 1～31 までの整数でなくてはなりません．〈年〉を指定するときは下 2 桁だけでもけっこうです．〈年〉，〈月〉，〈日〉はマイナス(－)か，スラッシュ(／)で区切ります．

```
A>date ↵
Current date is Thu 1987-12-24
Enter new date: 1967-6-23 ↵ ……………1980年よりまえの入力は受付けない.
Invalid date
Enter new date: 2001-01-01 ↵
A>date ↵
Current date is Mon 2001-01-01
Enter new date:↵ ……………↵だけ押すと，日付の変更はしない.
A>
```

図 5.9　DATE コマンド

● TIME コマンド

書式　　〈時〉：〈分〉：〈秒〉

DATE コマンドと同じように内蔵時計の時刻を設定します．引数を与えると時刻を設定してコマンド・レベルに戻り，引数なしなら時刻を表示して新しい時刻を尋ねてくるのも DATE コマンドと同じです．

最初に時刻を設定する際に，〈時〉を 0～23 の数字で設定すれば 24 時間制，0～12 の数字に A や P を付けて設定すれば 12 時間制で時刻を刻みます．また，〈時〉，〈分〉，〈秒〉はコロン(：)で区切って入力します(**図 5.10**)．

```
A>time ↵
Current time is 12:02:27.00a
Enter new time: 12:05:50 ↵ ……………何もつけないと，24時間制とみなされる.
A>time ↵
Current time is 12:05:52.00p
Enter new time:↵ ……………↵だけ押すと，時刻の変更はしない.
A>
```

図 5.10　TIME コマンド

● VERIFYコマンド

書式　VERIFY {ON | OFF}

このコマンドはディスクに書き込みを行う際(COPYなどを使った場合)に，きちんと書き込めたことをチェックするかどうかを決めるコマンドです．

ベリファイをONにすると，ディスクに書き込むごとにチェックをするので，信頼性は上がりますが時間も余計にかかります．ディスクはよほど安物を使わない限り，信頼してよいでしょうからあまり使う必要はありません．MSX-DOS起動時はOFFになっています．システムなど大切なファイルをコピーするときなどにONにするとよいかもしれません．

では，これらのコマンドを使ったイニシャライズ用(DOSを起動させた直後の初期設定を行う)バッチ・ファイルを作ってみましょう．**図5.11**のようになります．MODEで指定している画面表示文字幅は，使用しているコンピュータがMSXかMSX2かに合わせて，40か80にしてください．

```
A>copy con init.bat ⏎
date ⏎
time ⏎
mode 80 ⏎ ··············MSX2の場合はこのままでよいが，MSXでは
                        mode 40に変更する必要がある．
verify on ⏎
^Z ⏎
        1 file copied
A>
```

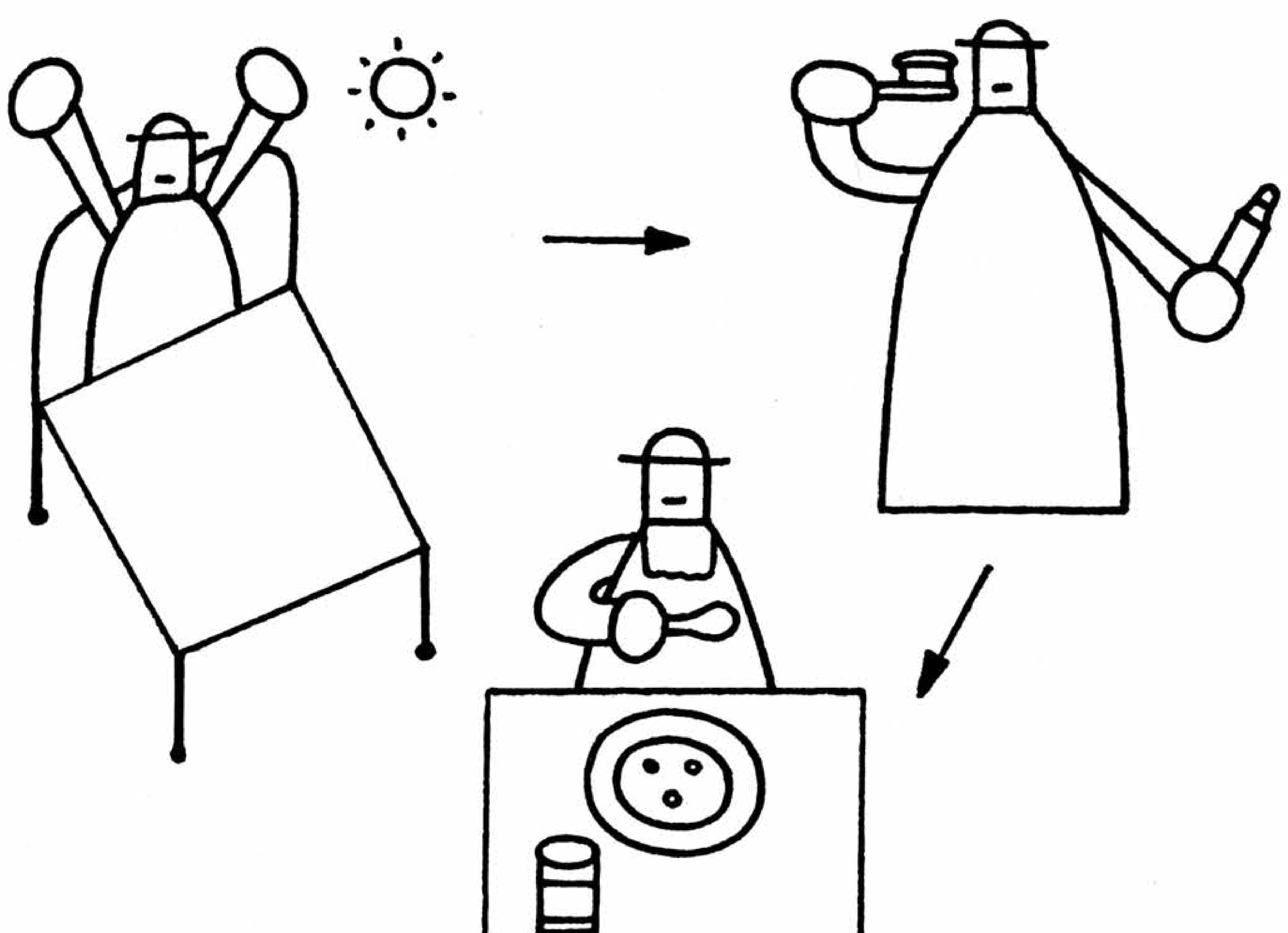
図5.11　イニシャライズ用バッチ・ファイル

このバッチ・ファイルをDOSが起動した直後に実行すると，日付と時刻を設定して，画面をクリアし，ベリファイ・モードONにしてからコマンド・レベルに戻ります．ベリファイはOFFにするのが普通ですが，ONにするとコピーがどのくらい遅くなるのかを実験してみるのもよいでしょう．

5-3 MSX-DOS起動時に実行されるバッチ・ファイル

前節では，DOSが起動するときにイニシャライズを行うバッチ・ファイルを作りました．このようなバッチ・ファイルはDOSを起動させたらかならず実行するべきものですからいちいちコマンド・レベルから呼び出さなくても，DOSの起動と同時に自動的に実行してほしいものです．

このような機能を実現するために，バッチ・ファイルには1つだけ特殊な意味を持つファイル名があります．それは，AUTOEXEC.BATです．このファイル名を持ったバッチ・ファイルは，普通のバッチ・ファイルとまったく同じように使えますが，ただDOSを起動させたときに自動的に実行されるという点が違います．



では，さきほどのバッチ・ファイルを**図5.12**のように，RENコマンドを使ってリネームしてください．

```
A>ren init.bat autoexec.bat ⏎
A>dir *.bat ⏎
EXCHANGE BAT       31 87-12-24  4:58p
AUTOEXEC BAT       33 01-01-01 12:06p
       2 files   717824 bytes free
A>
```

図5.12　INIT.BATをAUTOEXEC.BATにリネーム

いまAUTOEXEC.BATを作ったディスクはDOSを起動させなくてはなりませんから，システムを持っている必要があります．もし，MSXDOS.SYSとCOMMAND.COMがなかったら他のシステム・ディスクからコピーしておいてください．

準備が整ったらリセットをボタンを押してみましょう（リセット・ボタンのないMSXでは，一度電源を切ってから再び起動させてください）．

```
MSX-DOS version 1.03
Copyright 1984 by Microsoft

COMMAND version 1.11

A>date ·································· 自動的にAUTOEXEC.BATが
Current date is Mon 2001-01-01  実行されている．
Enter new date:
```

図5.13　DOSの再起動

図 5.13 のように，MSX-DOS が起動して自動的に AUTOEXEC.BAT が実行されます．もちろん，AUTOEXEC.BAT は普通にコマンド・レベルから呼び出して使うこともできます(**図 5.14**)．

```
A>autoexec ↵ ……………もちろんメイン・ファイル名を指定しても起動.
A>date
Current date is Mon 2001-01-01
Enter new date:
```

図 5.14　コマンド・レベルから AUTOEXEC.BAT を実行

このAUTOEXEC.BAT を使えば，DOS を起動したときに自分のメッセージを表示させることも可能です．

あらかじめ，**図 5.15** のようなテキスト・ファイルを作っておきます．

```
A>copy con message.doc ↵
Good morning, Dr.Chandra! ↵  ┐
↵                            │ 表示させたいメッセージを
I'm HAL-9000 Computer. ↵     ┘ ファイルにしておく.
^Z ↵
        1 file copied
A>
```

図 5.15　メッセージ用テキスト・ファイルの作成

このテキスト・ファイルを AUTOEXEC.BAT で表示するようにしておけば，DOS の起動と同時にこのメッセージが表示されるわけです．具体的には，AUTOEXEC.BAT の内容を **図 5.16** のようにしておきます．

```
A>copy con autoexec.bat ⏎
type message.doc ⏎    ……………TYPEコマンドでメッセージ・
^Z ⏎                        ファイルを表示させる.
        1 file copied
A>
```

図 5.16　AUTOEXEC.BAT の作成

さあ，リセットしてみましょう．**図 5.17** のようになるはずです．

```
MSX-DOS version 1.03
Copyright 1984 by Microsoft

COMMAND version 1.11

A>type message.doc
Good morning, Dr.Chandra!

I'm HAL-9000 Computer.

A>
```

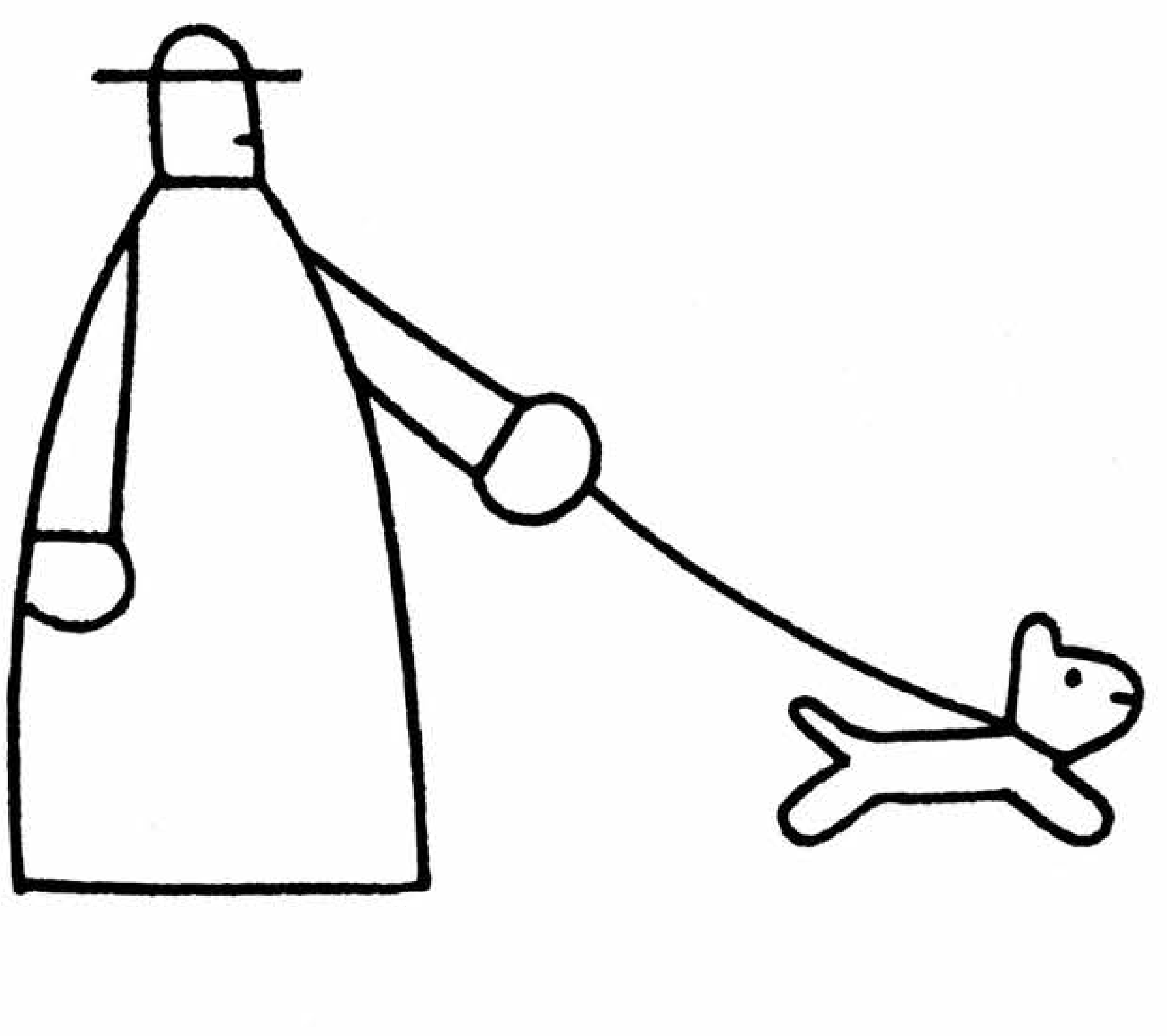
図 5.17　AUTOEXEC.BAT の実行

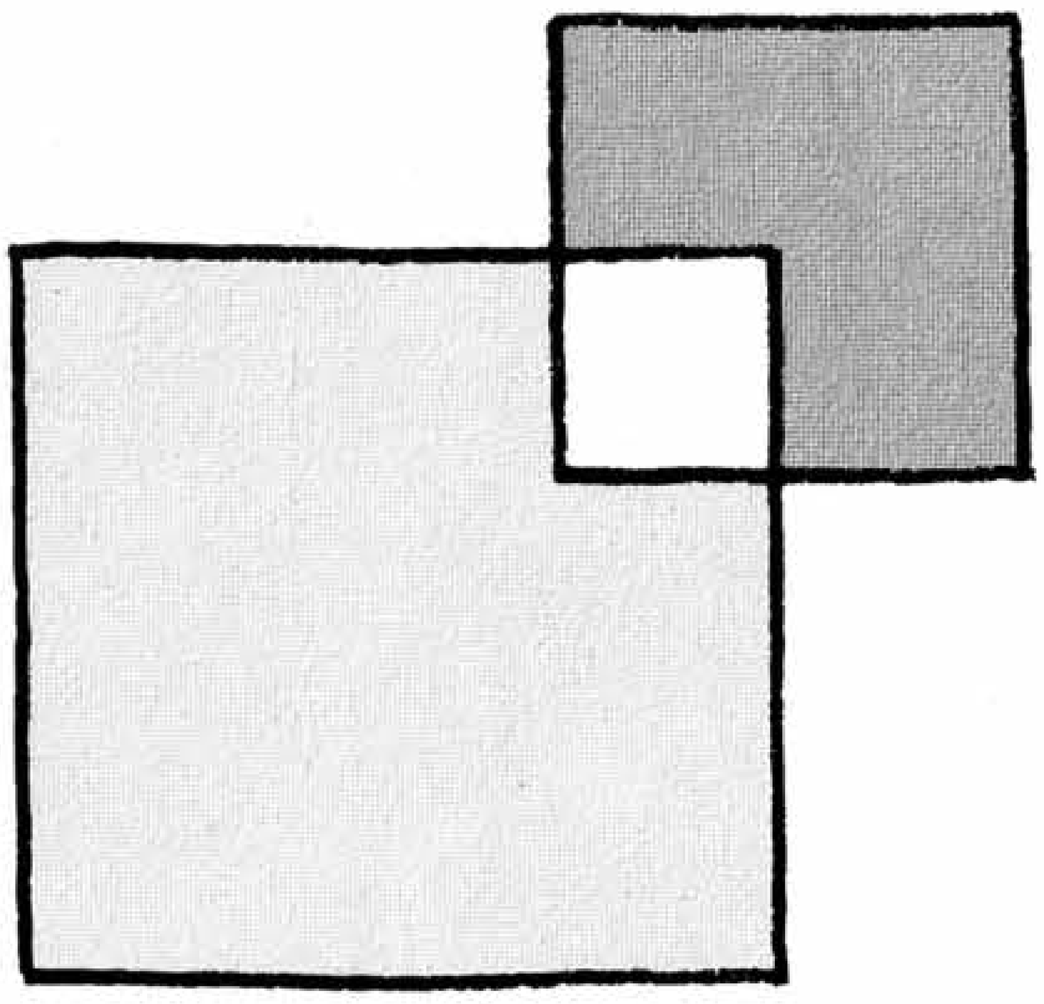
このようにDOSが起動すると同時にメッセージを表示するというのは，よくアプリケーション・ソフトウェアなどで使われる方法です．

6章

アプリケーションソフトウェアを使う

— ワープロ,ユーティリティ,ユーザー・インターフェイス —





6 1 MSX-DOS TOOLS

前章までで DOS の使い方についての説明は終わりました．これで，一通り DOS のコマンドを操作することができるようになりました．しかし，これまでに説明してきたことは，DOS 上に提供されるアプリケーション・ソフトウェアを使うための基礎知識だということを忘れてはなりません．1章でも述べたように，DOS の目的は，ユーザーがさまざまなソフトウェアをより手軽に使うための環境の実現なのです．DOS の目的を理解して使ってこそ，本当に MSX-DOS を使いこなしているということができます．そこで，本章と次章ではアプリケーション・ソフトウェアを紹介していきたいと思います．



写真 6.1　MSX-DOS　TOOLS

まず，一番最初に紹介するアプリケーション・ソフトウェアは「**MSX-DOS TOOLS**」(株式会社アスキー，以下 TOOLS と呼ぶ)です．このソフトは，さまざまな便利なコマンドの集合体であり，MSX-DOS の内部コマンドにはなかった高機能なコマンドも揃っています．MSX-DOS の活用には欠かせないソフトであり，アプリケーション・ソフトウェアというよりはシステムの一部と考えてもよいでしょう．

また TOOLS のコマンドでは，I／O リダイレクション，パイプ処理(後述)

と呼ばれる，MS-DOSのような入出力環境もサポートされ，大変使いやすいものとなっています．

ここで，TOOLSに含まれるコマンドの一覧表を**表6.1**にあげてみましょう．

コマンド名	機　　能
BEEP	BEEP音の発生
BIO	バイオリズムの出力
BODY	ファイルの一部の切出し
BSAVE	HEXファイルのバイナリファイルへの変換
CAL	カレンダーの表示
CALC	簡易電卓
CHKDSK	ディスクの状態の表示
CLS	クリアスクリーン
DISKCOPY	バックアップコピー及び照合
DUMP	ファイルの16進ダンプ
ECHO	コマンド行の表示
EXPAND	スペース・タブの変換
GREP	任意の文字列の検索
HEAD	ファイルの先頭部分の表示
HELP	コマンドリファレンスの表示
KEY	ファンクションキーの設定
LIST	BASICの中間言語ファイルのソース・ファイルへの変換
LS	ディレクトリの詳細情報の表示
MENU	ディレクトリのファイルの諸操作
MORE	ファイルの表示
PATCH	バイナリファイルの更新
SLEEP	アラーム機能
SORT	ソート(クイックソート)
TAIL	ファイルの末尾部分の表示
TR	文字列の置き換え
UNIQ	重複行の削除
VIEW	ファイルの表示(スクロール)
WC	語数，行数，ページ数カウント

表6.1　MSX-DOS　TOOLS　コマンド一覧

3章でBASICを使って作ったDUMPコマンドや，7章で作るECHOコマンドなどもTOOLSのコマンドには，最初から含まれています．また，TOOLSには，ここで紹介したほかにも便利なコマンドがあり，さらにM80，L80という本格的なアセンブラとリンカ，それにMEDというエディタまでも付属しています．M80，L80については，7章で実際にコマンドを作ることによって紹介することにします．本章では，これらの中でも頻繁に使うと思われるコマンドについてここで説明しておきましょう．詳しい説明はTOOLSのマニュアル，もしくはヘルプメッセージを参照してもらうことにして，ここでは主なコマンドの概略を説明していくことにします．

● CHKDSK　[／H] [／F] [／M]　〈ドライブ名〉

ディスクの使用状況(ディスク上のファイル状況，残り容量)を知らせてくれるコマンドです．スイッチにはヘルプ・メッセージ(使い方の説明)を表示する／Hと，破損クラスタ(ディスク上で傷が付いて使えなくなってしまった場所など)を修復する／F，さらにクラスタ，セクタ，ディレクトリ・エントリの数などのメディアに関する情報を表示する／Mの3つがあります．

いま，／Fスイッチは，破損クラスタを修復すると述べましたが，もちろん物理的に破損したディスクをプログラムで直すことはできません．単に，破損している部分に対してアクセスしないようにしておくだけです．これはその場しのぎですので，もし，ディスクに破損クラスタ発見されたら，新しいディスクにバックアップをとっておいたほうがよいでしょう．

また，／HはTOOLSの他のコマンドについても有効ですのでコマンドの使い方がわからなかったら／Hを付ければよいのです．

では，実際に使ってみましょう．**図6.1**では，スイッチを付けずに，ディスクの使用状況を表示させたのちに，／Mスイッチを付けてより詳しい情報を表示させています．

```
A>chkdsk ⏎

     713K total disk space          ┐
     350K in 46 user files          ├ ディスクの使用状況を表示.
     363K available disk space      ┘


A>chkdsk /m ⏎

     512  bytes per sector          ┐
                                    │
       1* first FAT sector          │
       3  sectors per FAT           │
       2  copies of FAT             │
                                    │ ／Mスイッチを指定すると,
       7* first directory sector    ├ メディアに関する情報も併せ
     112  directory entries         │ て表示.
                                    │
      14* first data sector         │
    1024  bytes per cluster         │
     713  total clusters            ┘

     713K total disk space
     350K in 46 user files
     363K available disk space


A>
```

図 6.1 CHKDSK コマンドの実行例

● DISKCOPY　[／H] [／V] [／C]　〈ドライブ名1〉　〈ドライブ名2〉

ドライブ名 1 で示されるディスクの内容をドライブ名 2 で指定されるディスクにすべてコピーします．／V スイッチは 2 枚のディスクの記録内容が同一であるか照合(ベリファイ)し，／C はコピーとベリファイの両方を行います．

このコマンドはディスクの内容をそのまま，まるごとコピーしますので，異なるメディア間(2DD と 1DD の間など)でのコピーはできません．

図 6.2 のようにコマンドを与えると，ドライブAの内容をドライブBのディスクにまるごとコピーしてから，ベリファイを行います．

また，DISKCOPY コマンドを実行すると，〈ドライブ名 2〉のディスクの内容は失われてしまいます．ですから普通はフォーマット直後のディスクに，ほかのディスクの内容をそのままコピーするときに使います．

```
A>diskcopy /c a: b:⏎  ··············ドライブAのディスクの内容を,
Set source disk on drive A:          まるごとドライブBにコピーする.
Set destination disk on drive B:
and strike any key when ready....
Complete
copy other disk (Y/N) ?n

A>
```

図 6.2　DISKCOPY コマンドの実行例

● DUMP　〈ファイル・スペック〉

ファイルの内容を 16 進およびキャラクタ・コードで表示します．3 章では，BASIC を使って作りましたが，より高機能になっています．このコマンドにもスイッチがたくさんありますが，複雑ですので省略します．

図 6.3 では，MSXDOS.SYS の 0050h から 009Fh の内容を表示させています．

```
A>dump /s0050 /e009f msxdos.sys ⏎  ··············MSXDOS.SYSの50h番地から
A:MSXDOS.SYS:                                     9Fh番地を表示.
0050: 03 FF 03 0D 04 10 04 14 04 1D 04 20 04 41 04 55  ...........  .A.U
0060: 04 5A 04 69 04 71 04 74 04 84 04 91 04 9C 04 9F  .Z.i.q.t........
0070: 04 14 05 17 05 1A 05 1D 05 20 05 2F 05 3B 05 02  .........  ./.;..
0080: 08 05 08 08 08 0B 08 0E 08 15 08 32 08 4A 08 00  ...........2.J..
0090: 00 79 FE 2C 20 09 D5 CD 75 12 43 D1 C3 75 2B 3A  .y., .ﾕﾍu.Cﾑﾃu+:

A>
```

図 6.3　DUMP コマンドの実行例

● GREP　"文字列"　〈ファイル・スペック〉

標準入力から任意の文字列を検索し，標準出力に出力します．このコマンドのスイッチも複雑ですので省略します．**図 6.4** の例では 7 章で制作する C のプログラム・リスト FC.C の中から〝for〟を含んだ行を検索しています(実際に実験したい人は，7 章を見て FC.C を入力するか，ほかの適当なテキスト・ファイルを使ってください)．

ここで，**標準入出力**という言葉が出てきたので説明しておきましょう．コマンドがデータを読み込んでくる入力先を標準入力といい，コマンドがデータを出力する出力先を標準出力といいます．普通，標準入力はキーボードに，標準出力はディスプレイになっています．

```
A>grep "for" b:fc.c ⏎  ……………ドライブBにあるファイルFC.Cから
                             〝for〟という文字列を検索.

      B:FC.C:
   19:  for(i = 1, j = 0; i < argc; i++){    ┐ 検索文字列の含ま
   45:          for(EVER counter++){        │ れている行番号と
   63:          for(EVER counter++){        │ 内容を表示.
   79:  for (; *buf != '¥0'; buf++)         ┘
    4 lines contain "FOR"

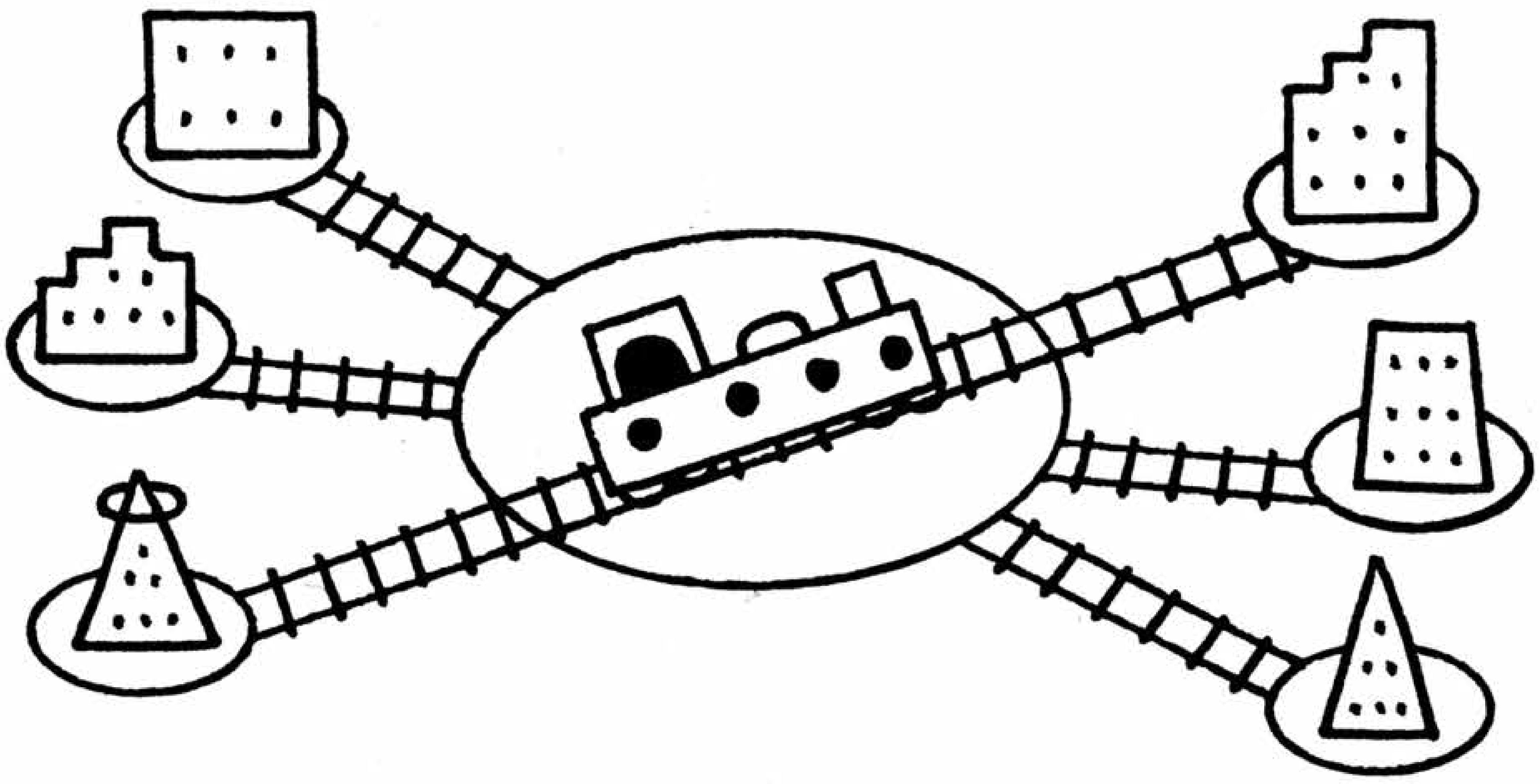
A>
```

図 6.4　GREP コマンドの実行例

● MORE　[／H] [／P〈行数〉]　〈ファイル・スペック〉

指定したファイルの内容を標準出力に 1 画面分だけ表示します．ファイルの内容が 1 画面分以上ある場合は，スペースキーもしくはリターンキーを押すまで表示が一時停止してくれます．ここで，スペースキーを押せば，次の 1 画面分，リターンキーならば 1 行分を表示し，再びキー入力待ちの状態になります．

／P スイッチによって，1 画面の出力行数を任意に変えることが可能です．TYPE コマンドでタイミングよく [CTRL] + [S] を押すのとほぼ同じ働きをしますが，標準入力から入力されたファイルの内容に対して行うことが可能



である点と，画面がスクロールするまえに自動的に表示が停止してくれる点でずっと使いやすくなっています．

さきほど，TOOLSでは，**I/O リダイレクション**をサポートしているといいました．ここで，このI/Oリダイレクションについて説明しましょう．I/Oリダイレクションとは，標準入出力を切り替える機能のことを指します．これにより，たとえばMOREコマンドでファイルを画面に出力する代わりにプリンタに出力したり，GREPやDUMPの結果を画面に出力するのではなく，ディスク上にファイルの形で出力させるといったことがことが可能です．これらの，標準入出力を切り替える先は，〝>〞，〝<〞という記号を使って指定します．この，I/Oリダイレクションの機能を使うことを，**リダイレクト**するといいます．

たとえば，さきほどやったFC.Cの中から〝for〞を検索した結果をプリンタに出力させたければ，GREPの結果をPRNにリダイレクトすればよいのです．

```
A>grep "for" fc.c > prn ⏎   ……………GREPコマンドの実行結果をデバイス・
                                  ファイルPRNにリダイレクト．
A>
```

図6.5 リダイレクト実行例

```
      A:FC.C:
19:   for(i = 1, j = 0; i < argc; i++){
45:           for(EVER counter++){
63:           for(EVER counter++){
79:   for (; *buf != '¥0'; buf++)
 4 lines contain "FOR"
```

図 6.6　リダイレクト実行結果

図 6.5 のように指定すると，図 6.6 のようにプリンタに結果が表示されます．同様に，GREP で文字列の検索を行い，その結果を RESULT.DOC というテキスト・ファイルにしたければ，図 6.7 のようにします．

```
A>copy con utmc.doc ↵
nue ↵
brain ↵
tek ↵
fug ↵
zdo ↵
pin ↵
tama ↵
mec ↵
^Z ↵
        1 file copied
A>grep "nue" utmc.doc > result.doc ↵

A>type result.doc ↵

         A:UTMC.DOC:
    1: nue
    1 lines contain "NUE"

A>
```

（copy con utmc.doc から ^Z まで）サンプルにするテキスト・ファイルの作成

（grep の行）…GREP コマンドの実行結果を RESULT.DOC というファイルにリダイレクト．

（type result.doc 以下）RESULT.DOC の内容を表示させると，GREP コマンドの実行結果がはいっている．

図 6.7　標準出力をディスク上のファイルにリダイレクト

このようにすれば，〝NUE〞という文字列がUTMC.DOCというファイルの中から検索されて結果がRESULT.DOCにリダイレクトされます．このRESULT.DOCは普通のテキスト・ファイルですので何回でもTYPEコマンドを使って表示させたり，エディタを使って編集することも可能です．

また標準出力のリダイレクト例ばかりあげましたが，標準入力のリダイレクトも存在します．例としてMOREの入力をFC.Cからリダイレクトしてみましょう．**図6.8**のようになります．

```
A>more < fc.c ⏎ ……………MOREコマンドへの標準入力をFC.Cにリダイレクト.
#include <stdio.h>
#define         YES     1
#define         NO      0
#define         EVER    ;;

main(argc, argv)
int     argc;
char    *argv[];

{
        int     i, j, counter = 0;
        int     flag1, flag2;
        FILE    *fp1, *fp2;
        char    buf1[255], buf2[255], *fn[2];
        char    caps = NO;
        char    ascii = YES;
        VOID    upstr();

        for(i = 1, j = 0; i < argc; i++){
                if (*argv[i] == '/') {
                        switch(toupper(*++argv[i])){
                                case 'C':       caps = YES;

==== More ? ====
```

図6.8 標準入力のリダイレクト

このように指定すれば，標準入力がFC.Cに切り替えられます．これは，**図6.9**のように指定するのとほとんど同じことです．

```
A>more fc.c ⏎ ……………この場合のFC.CはMOREコマンドの
                        引数としての意味を持つ

        A:FC.C:
#include <stdio.h>
#define         YES     1
#define         NO      0
#define         EVER    ;;

main(argc, argv)
int     argc;
char    *argv[];

{
        int     i, j, counter = 0;
        int     flag1, flag2;
        FILE    *fp1, *fp2;
        char    buf1[255], buf2[255], *fn[2];
        char    caps = NO;
        char    ascii = YES;
        VOID    upstr();

        for(i = 1, j = 0; i < argc; i++){
                if (*argv[i] == '/') {
                        switch(toupper(*++argv[i])){
                                case 'C':       caps = YES;

==== More ? ====
```

図 6.9　MORE コマンドの実行例

MORE がファイル名を引数としてとることができるので上の 2 つの例は同じ結果になるため，無理に使う必要はないのですが，MS-DOS などのフィルタとの互換性のために用意されています．

また，リダイレクトに似たものに**パイプ処理**があります．これは，あるコマンドの標準出力を次に実行するコマンドの標準入力につないでしまうものです．

たとえば，GREP の結果を MORE を使って見たいなどという場合には**図 6.10** のように指定します．

```
A>grep "nue" utmc.doc | more ⏎

        A:UTMC.DOC:
    1: nue
    1 lines contain "NUE"

==== More ? ====
```

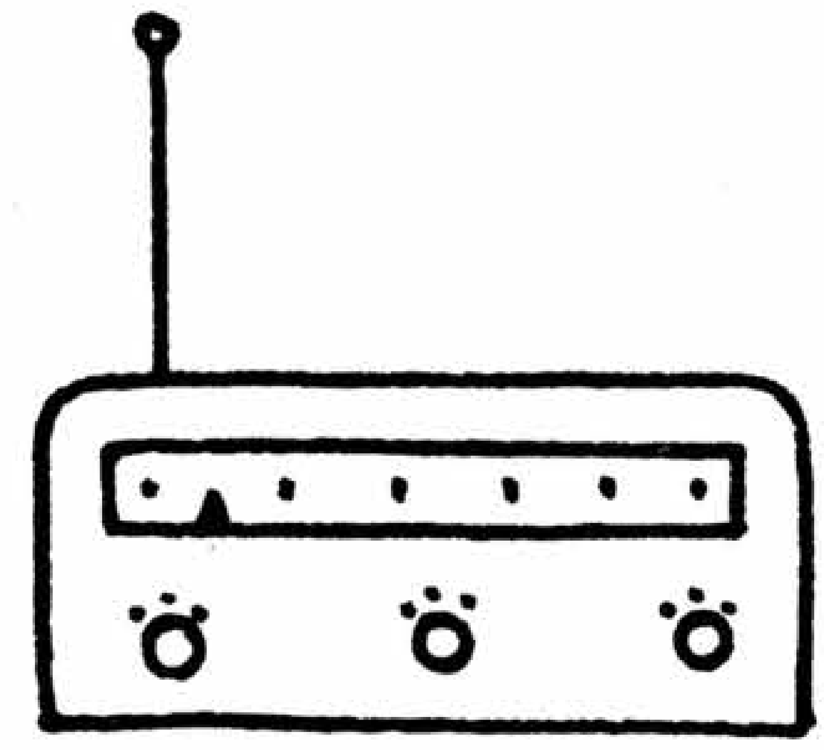
……………GREPコマンドの結果の出力が，そのままMOREコマンドの入力につながる．

図6.10　パイプ処理の例

〝|〟はパイプを示す記号で左のコマンドの結果を右のコマンドの標準入力につないでコマンドを実行します(リダイレクトではソースやデスティネーションはファイルでなくてはなりません．コマンドに標準出力をつなぐことはできません)．

リダイレクションやパイプは，MS-DOSなどでは最初からシステム側でサポートされていますが，MSX-DOSでは残念ながらシステムではサポートされていません．しかしTOOLSでは，各コマンドでサポートしているため，これらの機能を実現しています．これにより，コマンド操作が効率的に行えるばかりでなく，MS-DOSに近い操作性を味わうことができます．

TOOLSの紹介はこれくらいにしておきますが，MSX-DOSを本格的に使うにはTOOLSは欠かせません．ぜひ，揃えておきたいアプリケーション・ソフトウェアの1つです．



6.2 日本語MSX-Write



写真 6.2　日本語 MSX-Write

いまや，パーソナルコンピュータ上でもワープロが使えるのが当たりまえとなっています．日本のコンピュータなのですから，日本語を扱えなくては困るのですが，英文字と違って漢字は膨大な量があり，そのサポートも決して楽ではありません．そのため，MSX-DOS 単体では漢字機能はサポートされていません．しかし，漢字が使えないのでは日本で使う DOS としては役に立たないので，MSX には漢字を扱うためのアプリケーション・ソフトウェアが用意されています．それが，「**日本語 MSX-Write**」(株式会社アスキー，以下 MSX-Write と呼ぶ)です．MSX-Write は，カナ漢字変換機能を受け持つ**フロントエンド・プロセッサ MSX-JE** と，それを利用したワープロからなるアプリケーション・ソフトウェアです．MSX-Write を使えば，あなたの MSX もワープロに早変わりします．

MSX-Write 自体は ROM カートリッジの形で供給され，MSX-DOS の存在を意識することなしに使うことができます．

電源を切った状態で MSX-Write のカートリッジを差し込み，電源を入れると MSX-Write のスタート画面になります．ここで，適当なキーを押すと，**写真 6.3** のようなメインメニューが表示されます．

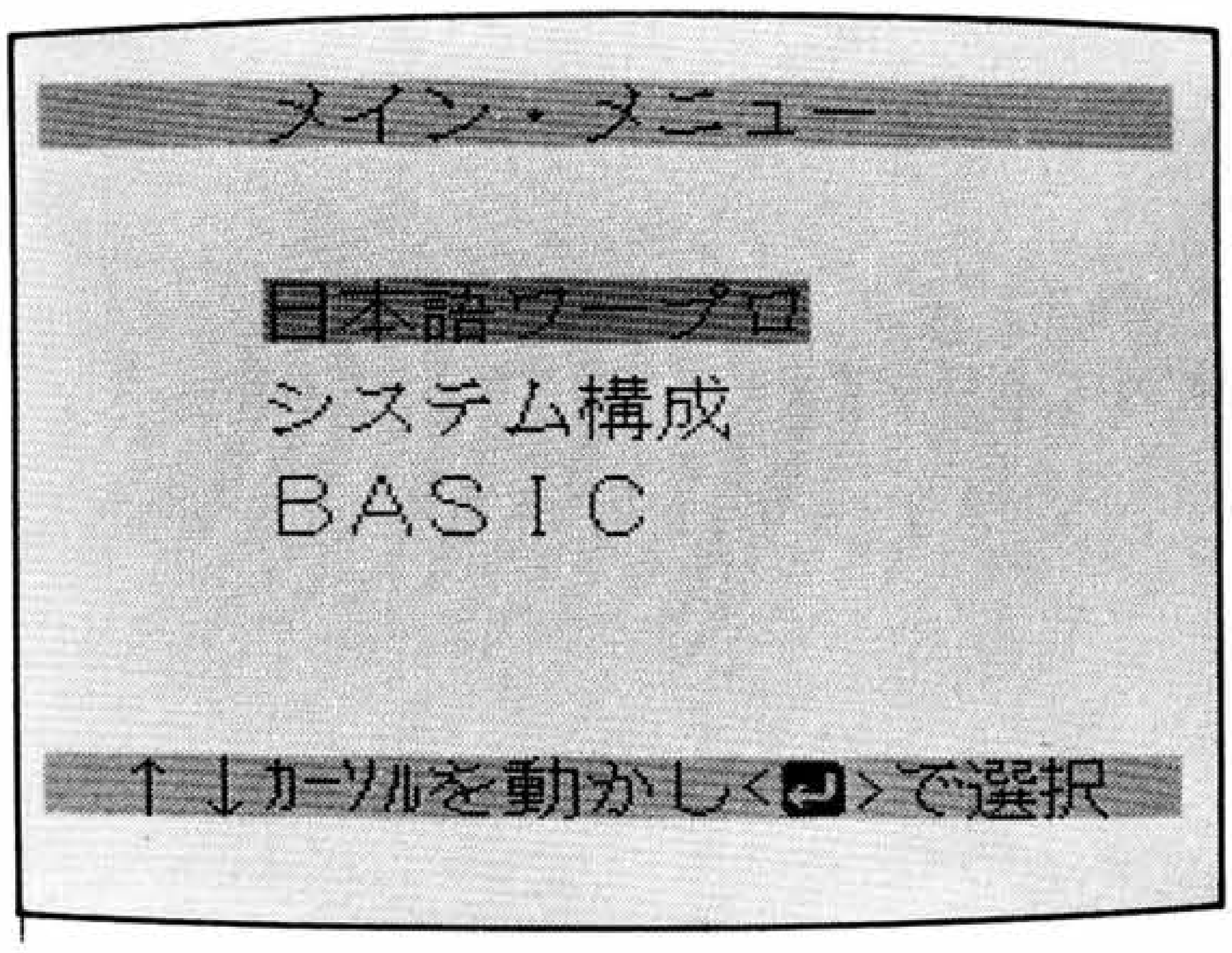


写真 6.3 メインメニュー

MSX-Writeを立ち上げたら, まず2番目のシステム構成を選んで, 自分の使っているMSXに合わせてください(**写真 6.4**).

これがすむと, 再びメインメニューに戻ってきます. 次に日本語ワープロを選んでみます. ここで, MSXとMSX2では画面の1行に表示できる漢字の文字数が15文字と30文字と違いますが, 基本的な操作は同じです. **写真 6.5** の状態になったら, もう普通のワープロとして使えます.



写真 6.4 システムの選択

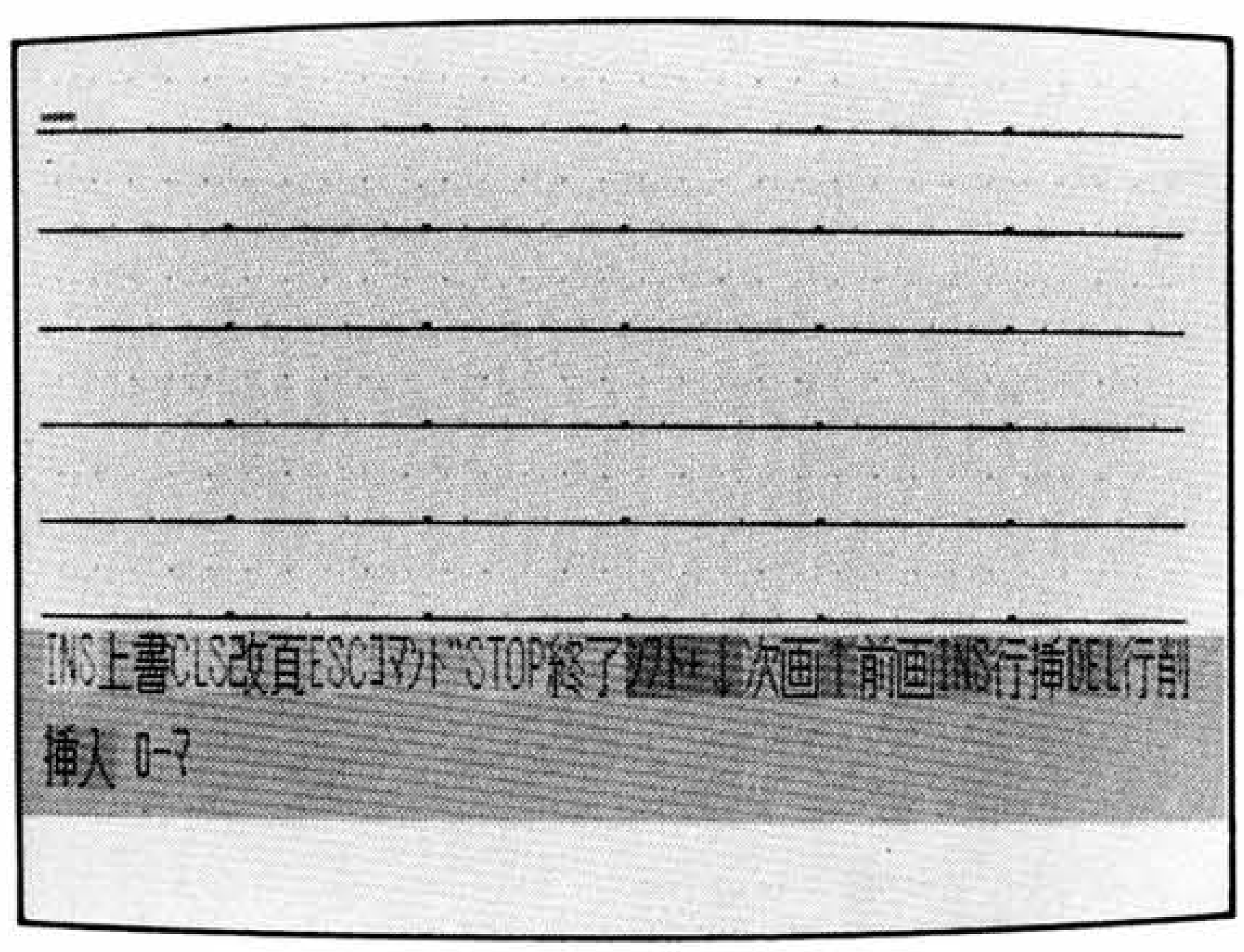


写真 6.5 ワープロ画面

ここでは, あまり詳しい説明はできませんが, スペースキーを変換キーとして使いながらカナ漢字変換がスムースに行えます. MSX-Writeは辞書をROMカートリッジ内に持っているので, カナ漢字変換は非常に高速です. まるで, MS-DOSなどの16ビットマシン上でRAMディスクを使っているかのような変換速度が味わえます.

この文書をディスク上に保存してみましょう．MSX-Write は MSX-DOS 上のアプリケーション・ソフトウェアですから MSX-DOS 用にフォーマットされたディスク上にデータを保存することができます．すでにフォーマットされたディスク（これまでの章で実習用に使っていたディスクでけっこうです）をセットしたら，CTRL ＋ F を押してファイルに関する機能を選択します．ここで，S を選択して文書をセーブすることを MSX-Write に指定します．文書全体をセーブするか，一部だけをセーブするのかを聞いてきますが，ここでは全体をセーブするので A を指定します．次に**写真 6.6** のようにファイル名を尋ねてきます．

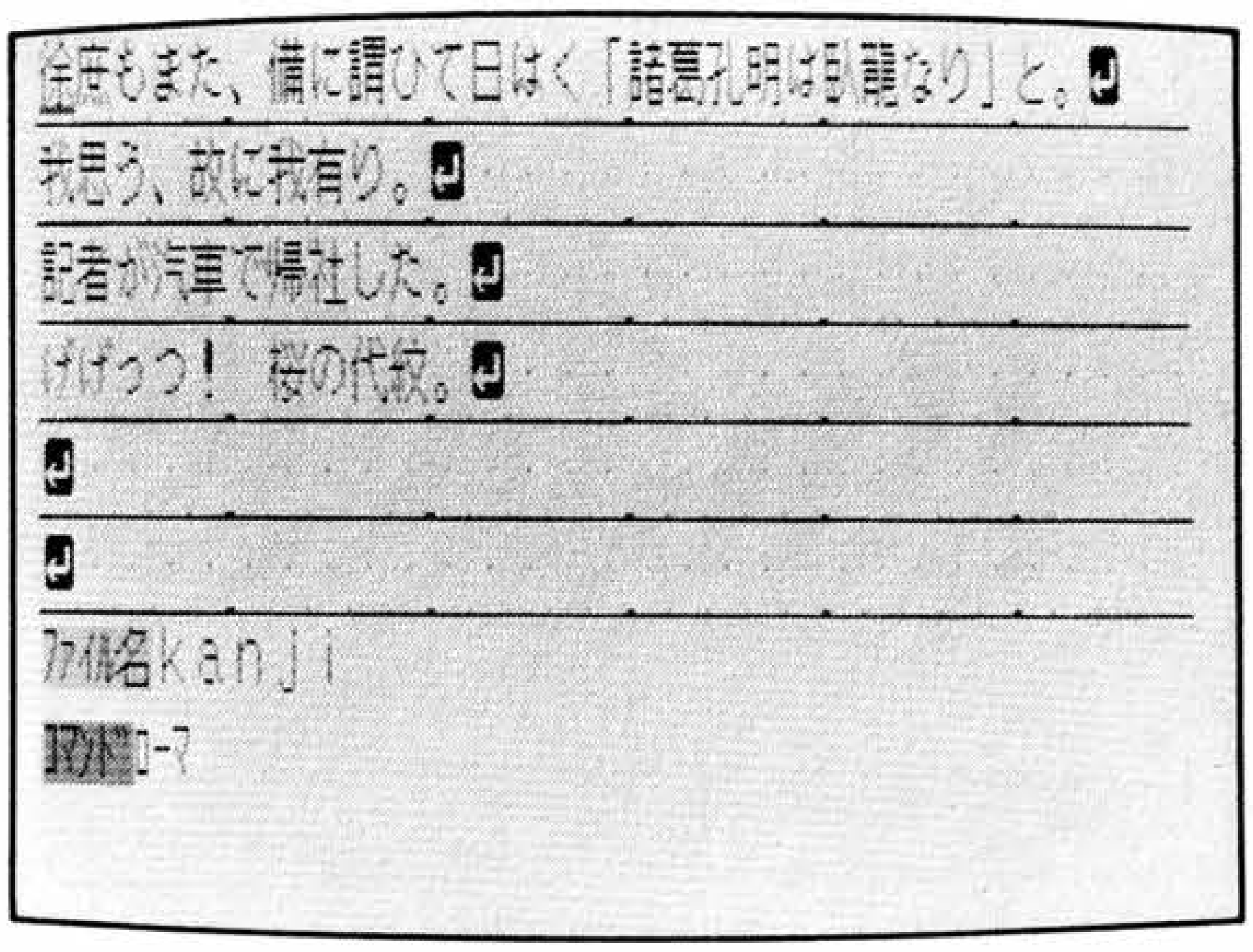


写真 6.6　ファイル名の指定

ここで，指定するファイル名はメイン・ファイル名のみで，拡張子は MSX-Write があらかじめ決まったものを付けてしまいます．

ディスクに書き込みがすんだら，STOP を押してメインメニューに戻ります．ドライブ A に MSX-DOS のシステムを含んだディスクをセットして，BASIC を選びます．このメニューでは BASIC となっていますが，システム・ディスクがセットしてあれば，BASIC ではなく MSX-DOS が起動します．

ここで，MSX-Write で作ったファイルのディレクトリを表示してみましょう．**図 6.11** のようになります．

```
A>dir kanji.* ⏎
KANJI    CTL      128 87-10-25  4:05p ……………補助ファイル
KANJI    TXT      128 87-10-25  4:05p ……………テキスト・ファイル
        2 files   645120 bytes free
A>
```

図 6.11　作成したファイルの確認

このように，KANJI.TXT と KANJI.CTL の 2 つのファイルができています．このうち，KANJI.TXT が実際に文字の情報のはいったテキスト・ファイルで，KANJI.CTL は画面制御関係のコードなどがはいった補助ファイルです．

KANJI.TXT は普通のテキスト・ファイルなのですが，TYPE コマンドで表示しても**図 6.12** のように漢字は表示されません．

```
A>type kanji.txt ⏎
っけっょ♣た♣ワ♣ス♥Aえゆ♣ノぃた♣ミ♣トうれ♣ヘ♣ユ♥uっええぉゅEかセ♣へぅぬきエ♣ネ♣ね♥v♣こ♥B
ぅとょv♣、♥Aャフ♣ノぅときL♣ね♥B
ぉLょメ♣エぉDょヤ♣ナぉAょミ♣オ♣ス♥B
♣ー♣ー♣チ♣ツ♥I♥@ゅら♣フあてかと♥B

A>                                        漢字が表示されない.
```

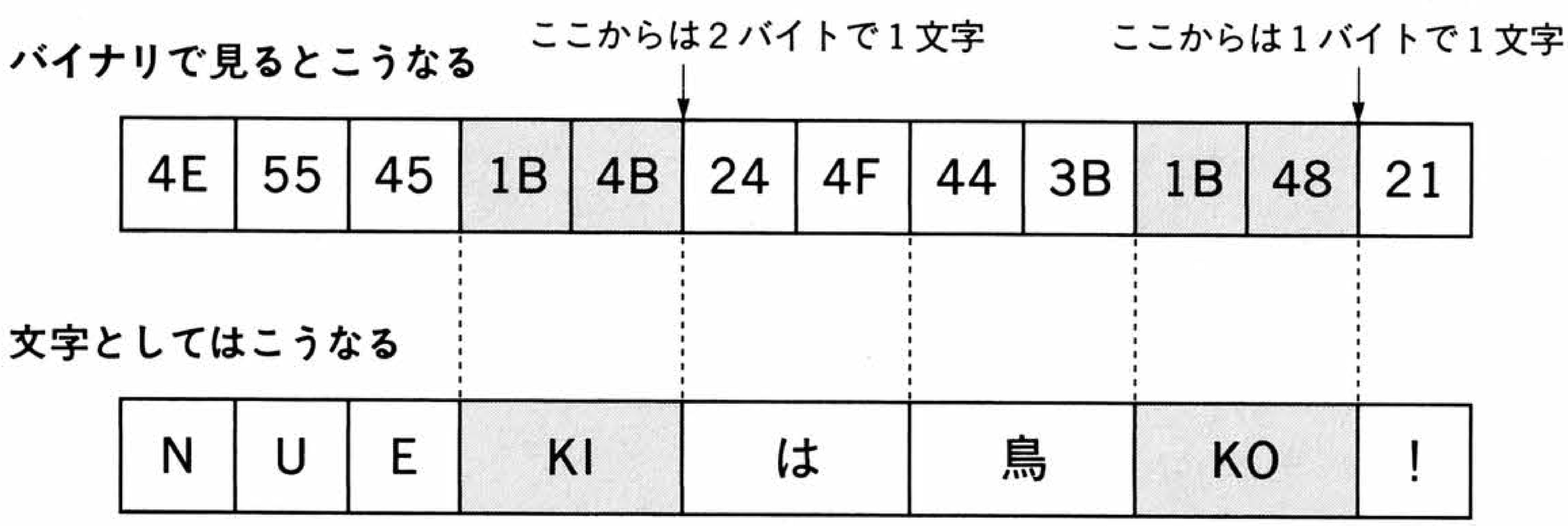
図 6.12　漢字のファイルを TYPE コマンドで表示

これは，MSX-DOS の TYPE コマンドが漢字の表示をサポートしていないためです．

ここで，漢字を表現するコードについて簡単に説明しましょう．普通の文字(英数字など)を表すコードには，ASCII コードや JIS コードなどがあり，MSX では JIS コード準拠のキャラクタコードを使用しています．キャラクタ・コードは，普通は 1 バイトで 1 文字を表します(MSX ではグラフィック・キャラクタは例外で 2 バイトで 1 文字を表します)．

1バイトで，0～255までの数を表現できますが，これでは256文字までしかコードを割り当てることができません．英数字だけならばこれで十分ですが，漢字を使おうと思うとこれではまったく足りません．そこで，2バイトを組にして漢字を表現することにしましょう．これならば，65536文字分のコードが用意できますから，漢字を扱うにも十分です．しかし，2バイトの数字の並びを見て，これが漢字1文字なのか，普通の英数字が2文字なのかを区別するにはどうしたらよいでしょうか？

まず思い付くのは，漢字を表すコードのまえには「これからあとは漢字である」ということを示す特別なコード(KIと呼ばれます)を置き，漢字を表すコードが終わったら「これで漢字のコードは終わりである」というコード(KOと呼ばれます)を置くという方法です．この考えで割り当てられた漢字コードをJIS漢字コードといいます．



普通の英数字(ANK文字)と，漢字(全角文字)を区別するために，JISコードでは，KI(KANJI IN)，KO(KANJI OUT)と呼ばれるコードで，はさまれた部分を漢字とみなす．

図6.13　JISコード概念図

この方法は，コードを単に順番に割り当てていけばよいので，人間にとっては理解しやすいといえます．そのため，MSX-Writeでも特殊コードの入力にはJISコードを使って入力します．しかし，テキスト・ファイルの内容をJISコードで表すにはいろいろ不都合があります．たとえば，テキスト・ファイルがKIとKOの分だけ大きくなってしまうことや，テキスト・ファイルの連結などの際に不必要なKI，KOを取り去ったり，テキスト・ファイルを分割する際にはKIやKOを付け加えたりしなくてはなりません．

これでは，ファイルを自由にコピー，連結，分割ができなくなってしまいます．そのために，MSX-DOSでは**図6.14**のようなシフトJISコード（正確には，シフトJISコードに手を加えたMS漢字コード）と呼ばれる方法で漢字を取り扱っています．シフトJISコードでは，漢字の最初の1バイト目を，普通は使わない80h～9Fh，E0h～FFhの内のどれかに決めてしまうのです．このためKIやKOなどの余分なコードは必要としませんし，漢字か普通の文字かどうかの判別も簡単に行えます．また，この方法はMS-DOSでも採用しています．

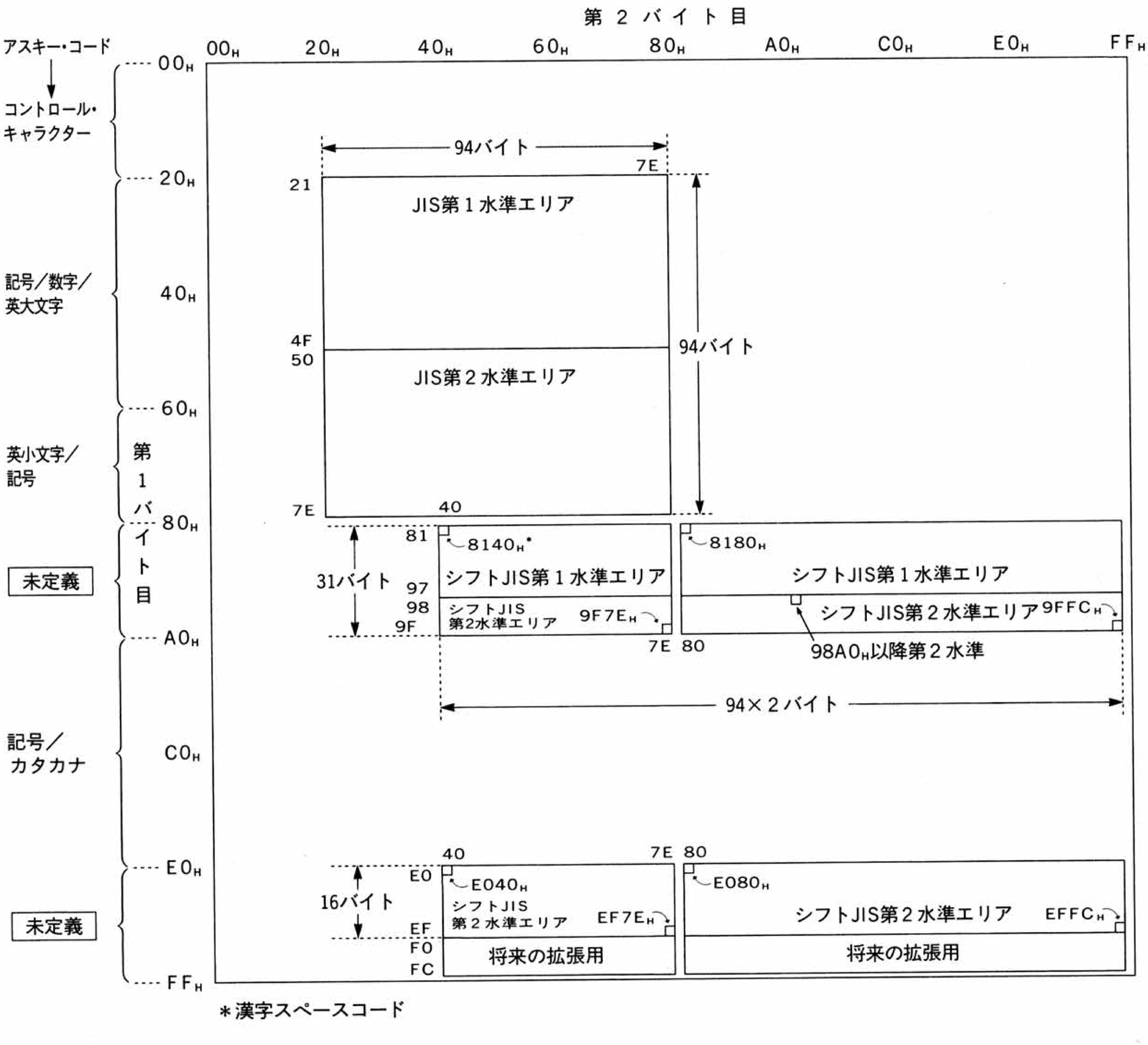


図6.14　JISとシフトJISの配置図

さて, MSX-Write を使えば漢字をシフト JIS で表したテキスト・ファイルを作ることができます. このファイルは, MS-DOS 上の TYPE コマンドを使えば, ちゃんと漢字で表示されます.

しかし, いくら MSX-DOS 上でサポートされていないからといってこの漢字を含むファイルを表示するのに, ほかの DOS を使ったり, MSX-Write を起動させなくてはならないのではあまりにも不便です.

本当はマシン語を使って, 漢字まで表示することのできる TYPE コマンドを作ればよいのですが, 大変ですから, ここでは BASIC を使って漢字を含むファイルを表示する KTYPE を作てみましょう. ただし, このプログラムは MSX2 でしか動きませんのでご注意ください.

リスト 6.1 のプログラムはアスキー・システムバンク『MSX2 BASIC 入門』の 6 章にあるプログラムを改造したものです. プログラムの詳しい説明については, そちらを見ていただくことにして, ここではとりあえず使い方だけを説明しましょう.

```
100 'KTYPE
110 CLS
120 MAXFILES = 2 : DEFINT A-Z
130  SC = 6 : MX = 496         '32 * 20 characters
140 'SC = 5 : MX = 240         '16 * 20 characters
150 IF SC = 6 THEN WIDTH 80 ELSE WIDTH 40
160 INPUT "file neme";FI$
170 OPEN "grp:" AS #1
180 ON ERROR GOTO 680
190 OPEN FI$ FOR INPUT AS #2
200 SCREEN SC,,,,3 : COLOR 3, 1, 0
210 COLOR = (1, 2, 4, 3) : COLOR = (2, 7, 0, 0) : COLOR = (3, 7, 7, 7)
220 GOSUB 630
230 'main loop
240 IF EOF(2) THEN 310
250 C$ = INPUT$(1, #2) : A = ASC(C$)
260 IF A < &H20 THEN GOSUB 400 : GOTO 240
270 IF (A < &H81) OR ((A > &H9F) AND (A < &HE0)) THEN GOSUB 440 : GOTO 240
280 IF EOF(2) THEN 310
290 SJ$ = C$ + INPUT$(1, #2) : GOSUB 480
300 GOTO 240
310 'end of file
320 X = 0 : GOSUB 580
330 C$ = "[EOF]" : COLOR 3, 2 : GOSUB 430
340 DUMMY$ = INPUT$(1)
```

```
350 SCREEN 0,,,,,0 : COLOR = NEW : COLOR 15, 4
360 CLOSE
370 IF SC = 6 THEN WIDTH 80 ELSE WIDTH 40
380 CALL SYSTEM
390 'CTR-codes
400 IF A = &HD THEN X = 0
410 IF A = &HA THEN GOSUB 580
420 RETURN
430 'ANK characters
440 SET PAGE ,0 : PRESET(X, Y) : PRINT #1, C$;
450 SET PAGE ,1 : PRESET(X, Y) : PRINT #1, C$;
460 X = X + 8 : IF X > MX THEN X = 0 : GOSUB 580
470 RETURN
480 'KANJI
490 HI = ASC(SJ$) : LO = ASC(RIGHT$(SJ$, 1))
500 HI = 2 * (HI + &H70*(HI < &HA0) + &HB0*(HI >= &HA0))
510 LO = LO + (LO >= &H80)
520 HI = HI + (LO < &H9E) : LO = LO + &H5E*(LO >= &H9E) - &H1F
530 CODE = HI * &H100 + LO
540 SET PAGE ,0 : PUT KANJI (X, Y), CODE,,,1
550 SET PAGE ,1 : PUT KANJI (X, Y), CODE,,,2
560 X = X + 16 : IF X > MX THEN X = 0 : GOSUB 580
570 RETURN
580 'CR and More
590 Y = Y + 10 : IF Y < 200 THEN RETURN
600 C$ = "==== More ? ====" : COLOR 3, 2 : GOSUB 430 : COLOR 3, 1
610 DUMMY$ = INPUT$(1)
620 GOSUB 630 : RETURN
630 'cls
640 SET PAGE 0, 0 : CLS
650 SET PAGE 1, 1 : CLS : PUT SPRITE 0, (0, 216), 0
660 X = 0 : Y = 0
670 RETURN
680 'error trap
690 PRINT "No file name!"
700 PRINT "Hit any key."
710 RESUME 340
```

リスト 6.1　KTYPE.BAS

```
basic ktype.bas
```

リスト 6.2　KTYPE.BAT

漢字をスクリーンに 32×20 だけ表示するか 16×20 にするかで，スクリーンモードが変わってきますので 130 行と 140 行のいずれかを REM 文にして使用してください．また，ファイル名はそれぞれ KTYPE.BAS，KTYPE.BAT としてください．

コマンド・レベルから〝KTYPE〟と入力すれば，**リスト 6.2** のバッチ・ファイルが BASIC を呼び出し，KTYPE.BAS を実行します．するとファイル名を聞いてくるので，画面に表示したいファイルのファイル名を入力すれば，漢字が表示されます．この KTYPE は MORE の機能も持っているので，画面いっぱいになると表示を一時停止してくれます．

490 行からの 4 行で，シフト JIS コードから JIS コードへの変換を行っています．BASIC の PUT KANJI 命令は，JIS コードで漢字を指定しなくてはならないためです．

この KTYPE を使って KANJI.TXT を表示すると次のようになります．

```
徐庶もまた、備に謂ひて曰はく「諸葛孔明は臥龍なり」と。
我思う、故に我有り。
記者が汽車で帰社した。
げげっつ！　桜の代紋。
```

図 6.15　KTYPE 実行結果

また，ここでは詳しくは触れませんが，MSX-Write は MSX-JE というフロントエンド・プロセッサを含んでいます．MSX-JE は他のアプリケーション・ソフトウェアが MSX-Write の強力なカナ漢字変換機能を使えるように用意されたプログラムです．アプリケーション・プログラムによってはこの MSX-JE を使って，MSX-Write 同様の快適な漢字環境が使用できるものもあります．

8 ビット・マシンと 16 ビット・マシンでは，漢字機能の点でどうしても差が付いてしまうものですが，このように MSX-Write を使えば MSX でも十分漢字を使うことが可能なのです．

6 3 ユーザーインターフェイス HALNOTE

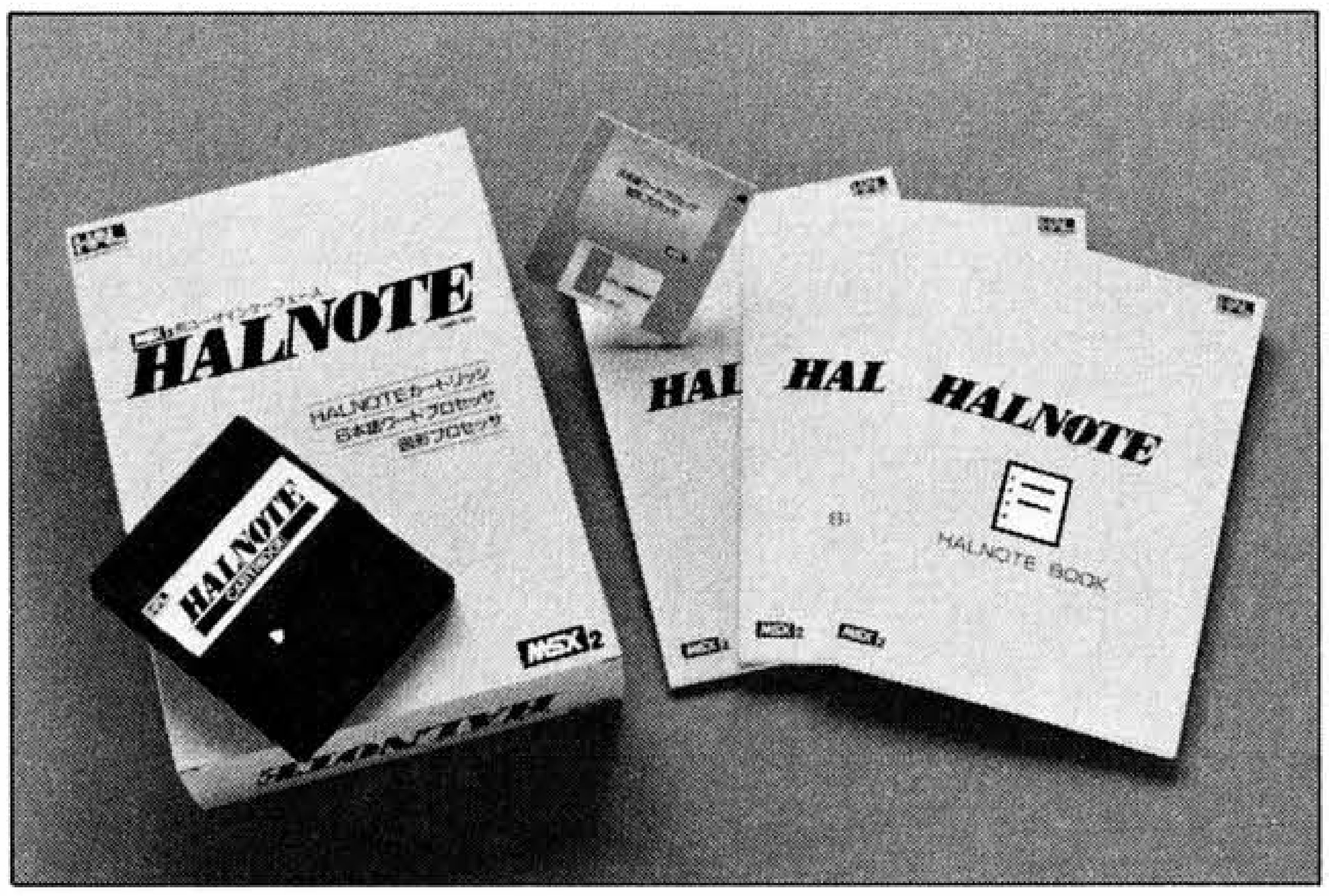


写真 6.7　HALNOTE

「**HALNOTE**」(株式会社 HAL 研究所) は MSX 上で動く**統合化ソフト**です. 統合化ソフトとは, よく使われるソフトウェアをひとつの環境内で使えるようにし, データの互換性を持たせたもので, HALNOTE では, **HALNOTE バインダ**と呼ばれるファイルマネージャと日本語ワードプロセッサ, 図形プロセッサを1つにまとめてあります.

HALNOTE バインダ

DOS と密接な関係があるのは, HALNOTE バインダです. バインダは, ひとことでいえば, MSX-DOS のコマンドを視覚的にとらえながらファイル管理をするソフトです. さらに, マウスを使えば, コマンドやファイルの指定の際に, わざわざキーボードから入力しなくてもメニューをマウスで指し示すだけですみますので, ファイルの管理が非常に簡単明瞭になっています.

実際の使用例を見てください. **写真 6.8** の四角い箱(セルといいます)の中にファイル名と**アイコン**と呼ばれる絵が出ています. このアイコンは, ファイルの種類を視覚的に表すもので, 自分で好きなものを付けることができま

す．もし，用意されているアイコン以外のものが使いたければ，自分でデザインすることも可能です．つまり，アイコンは，拡張子を視覚化したような働きを持っています．ただし，HALNOTE は，MSX-DOS 上のソフトですので拡張子も取り扱うことができます．アイコン表示から拡張子の表示に切り替えると，**写真 6.9** のようになります．

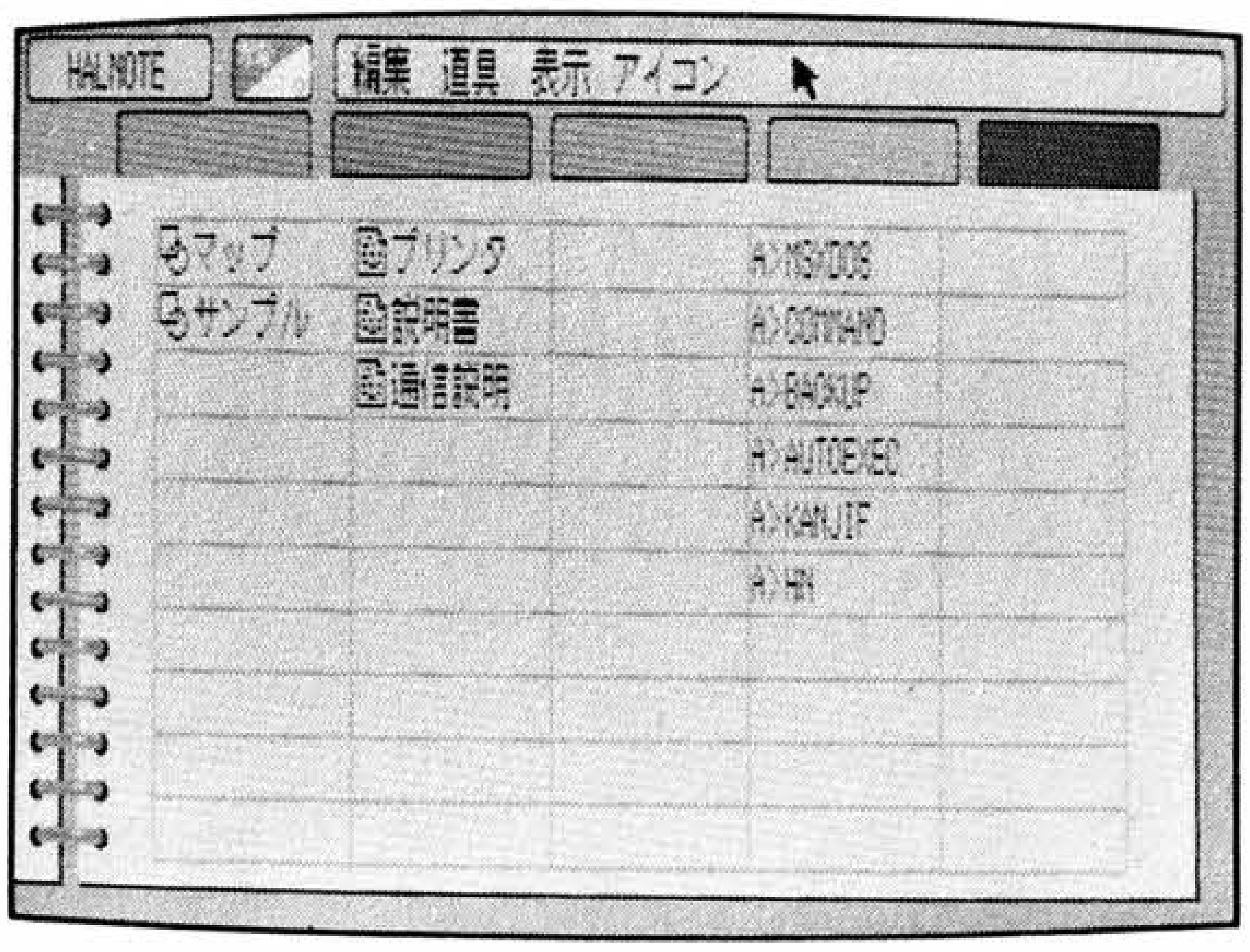


写真 6.8　バインダ(アイコン ON)

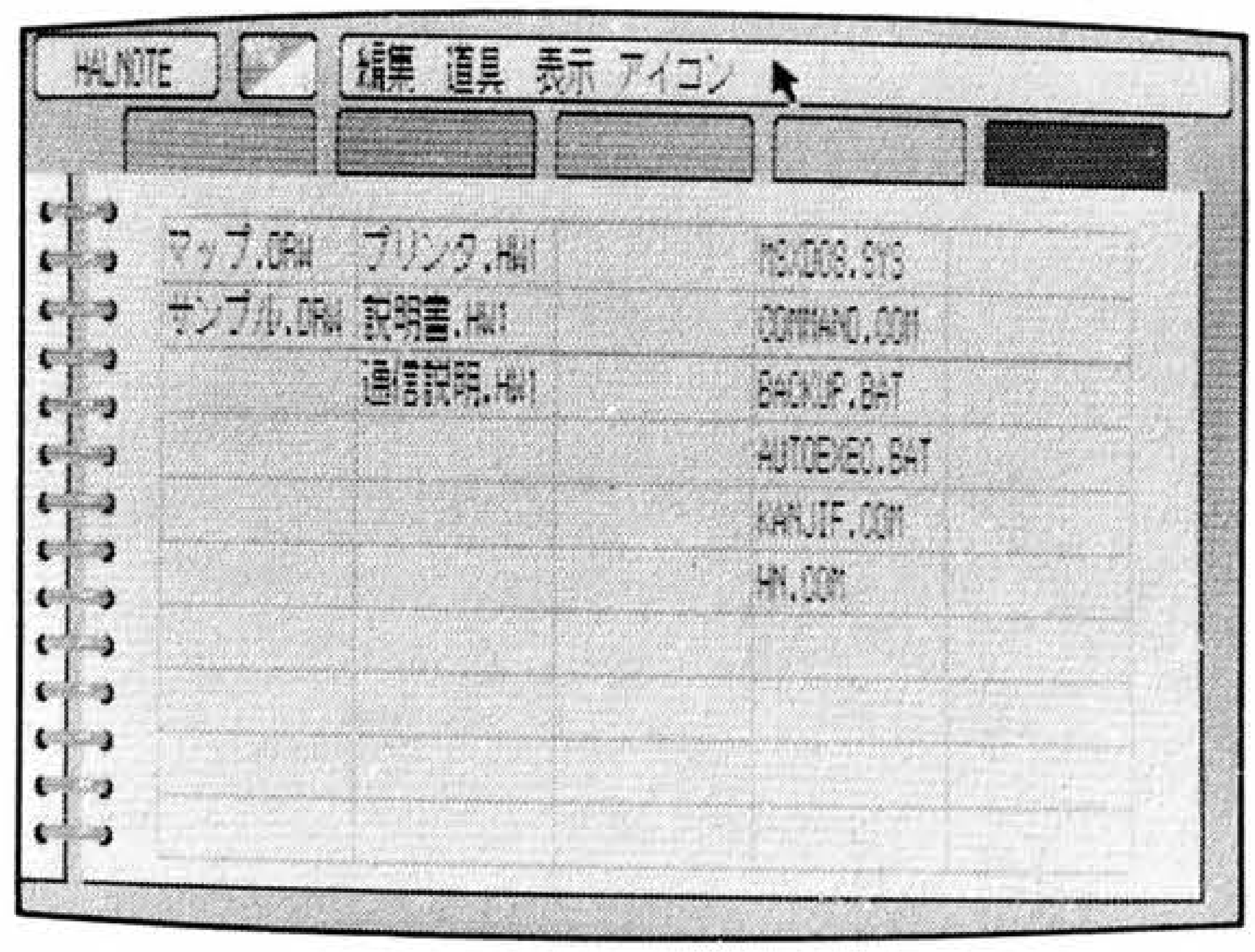
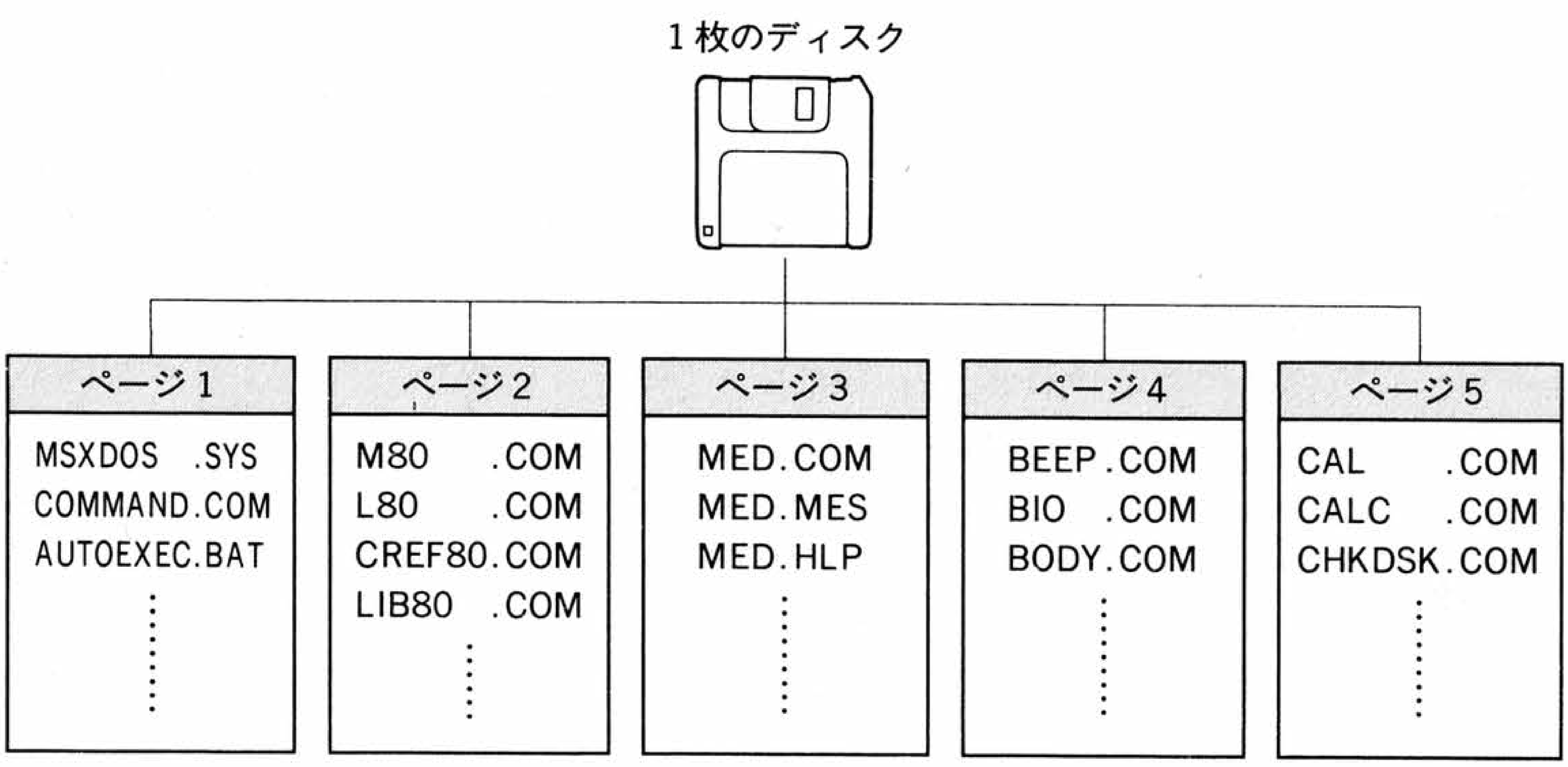


写真 6.9　バインダ(アイコン OFF)

このように，それぞれのファイルは拡張子もきちんと持っていることがわかります．



ページ1～5を，目的に応じて使い分けることによって，ディスクを効率よく使用できる．

図 6.16　HALNOTE バインダのページ概念図

このセルの一覧表がMSX-DOSのディレクトリを表示させることに相当しています．ただし，HALNOTEにはページという概念があり，一枚のディスクを5つの部分に分けて管理することができますから，この画面だけですべてのファイルが見られるわけではありません(**図6.16**)．

写真6.10は，ほかのページに切り替えてみたものです．

このページを分けたファイル管理は，使ってみると大変便利です．たとえば，ワープロなどを使っている場合に，作成した文書を書類，手紙，メモなど，目的別にページを決めてセーブしておくことにより，ディスクの中を整理することができます．

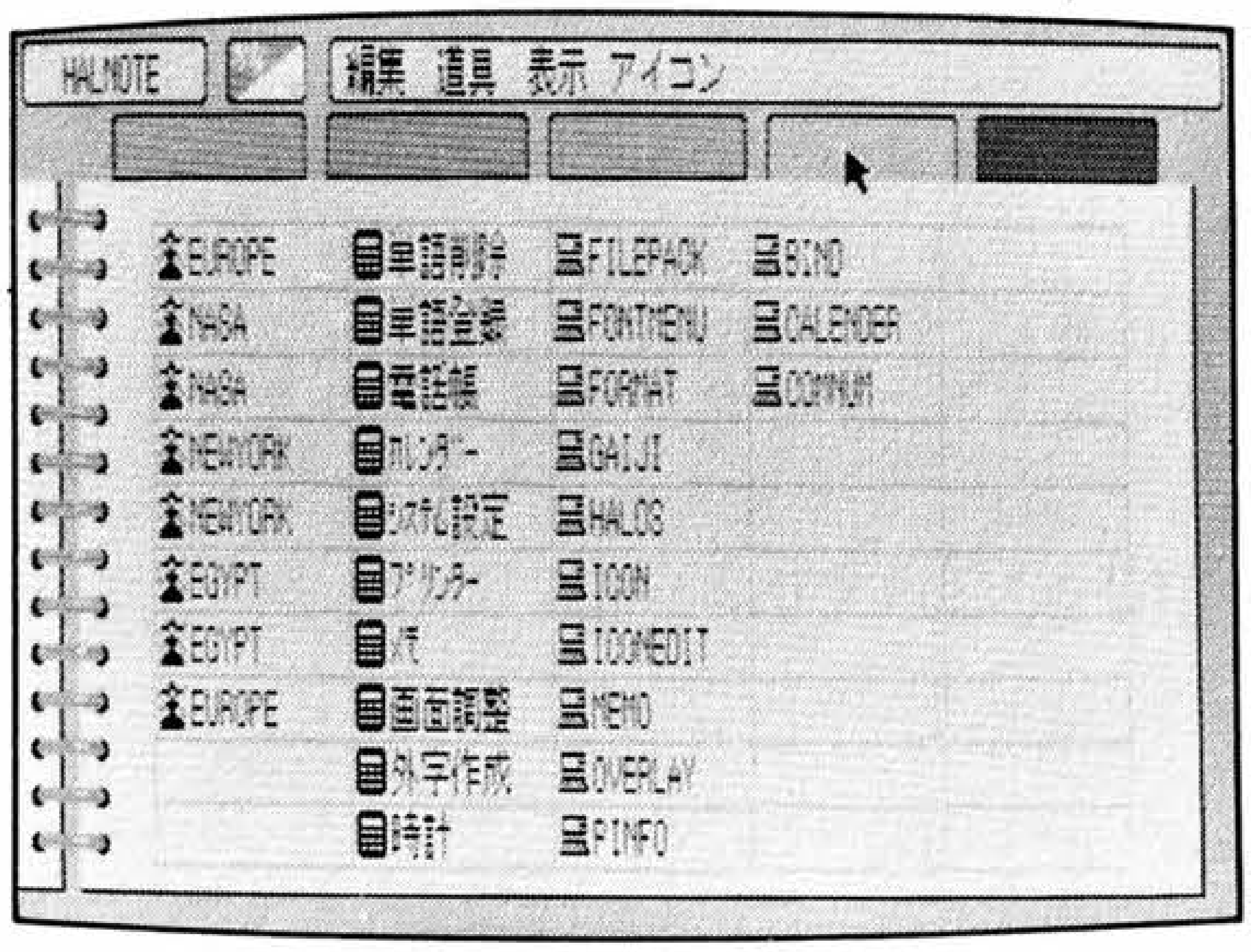


写真6.10　ページの変更

さて，ここでHALNOTEのファイル管理の一例として，ファイルの移動について見てみましょう．同じドライブ内でファイルの移動を行うのは簡単です．移動したいファイルのセル上にマウス・カーソルをもっていきマウスのボタンを2度クリック(軽く押すこと)してから，移動先のセル上にマウス・カーソルを移動させ，再びボタンをクリックするだけです(**写真6.11**)．

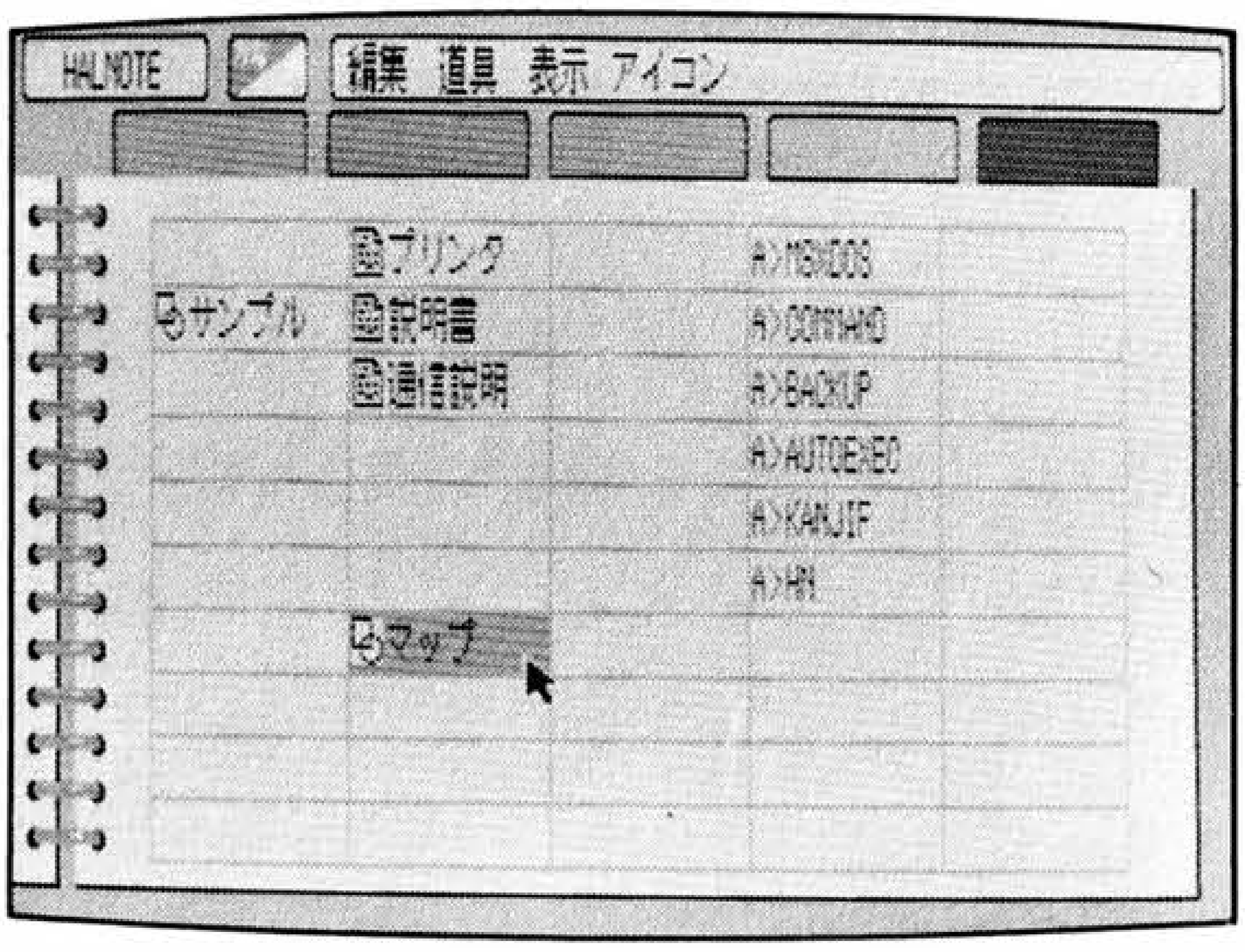


写真6.11　ファイルのページ内移動

もちろん，ファイルをドライブ間で移動させることもできます．移動したいファイルをマウスで指定してから，メニューの［編集］をマウスで指定すると，［編集］内のコマンドのメニューが表示されます．ここで，［複製(→B)］を選べばコピーが実行されます．

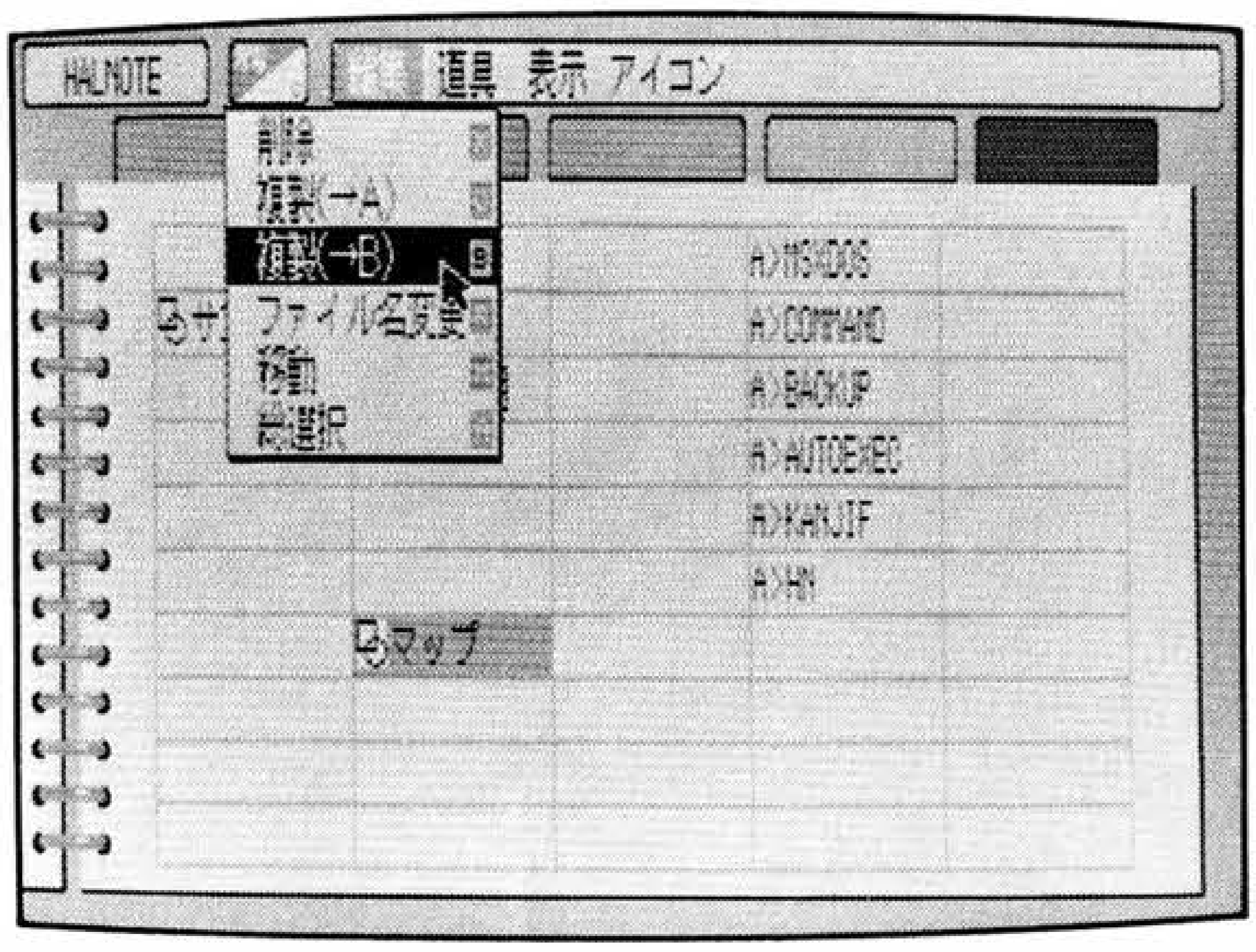


写真 6.12　ファイルの複製

写真 6.12 での操作は，〝COPY　COMMAND.COM　B:〟と同じことですが，このようにマウスを使ってファイル管理を行うならば，キーボードに不慣れな人にも容易に行えます．また，キーボードに熟達した人でも，ファイルを視覚的にとらえられるというのは，大きなメリットでしょう．

ファイルの移動以外にも，バインダ上では，ファイルの削除，ファイル名変更などが同様の操作で行えます．

また，HALNOTE 上では日本語ワードプロセッサの漢字入力機能が常に使用できるので，漢字のファイル名なども作成できます．

ただし，これらの漢字ファイル名は，HALNOTE 上で使用している分には，何の不都合もありませんが，MSX-DOS のコマンド・ラインからディレクトリを表示させた場合などには，わけのわからない文字に化けてしまい，読めなくなるので注意が必要です．

また，統合化ソフトは，デスクアクセサリーと称して，時計，カレンダ機能を備えているのが普通ですが，HALNOTE もやはり，時計，カレンダ，電話帳機能などをいつでも自由に画面上に呼び出し，使用することができます．

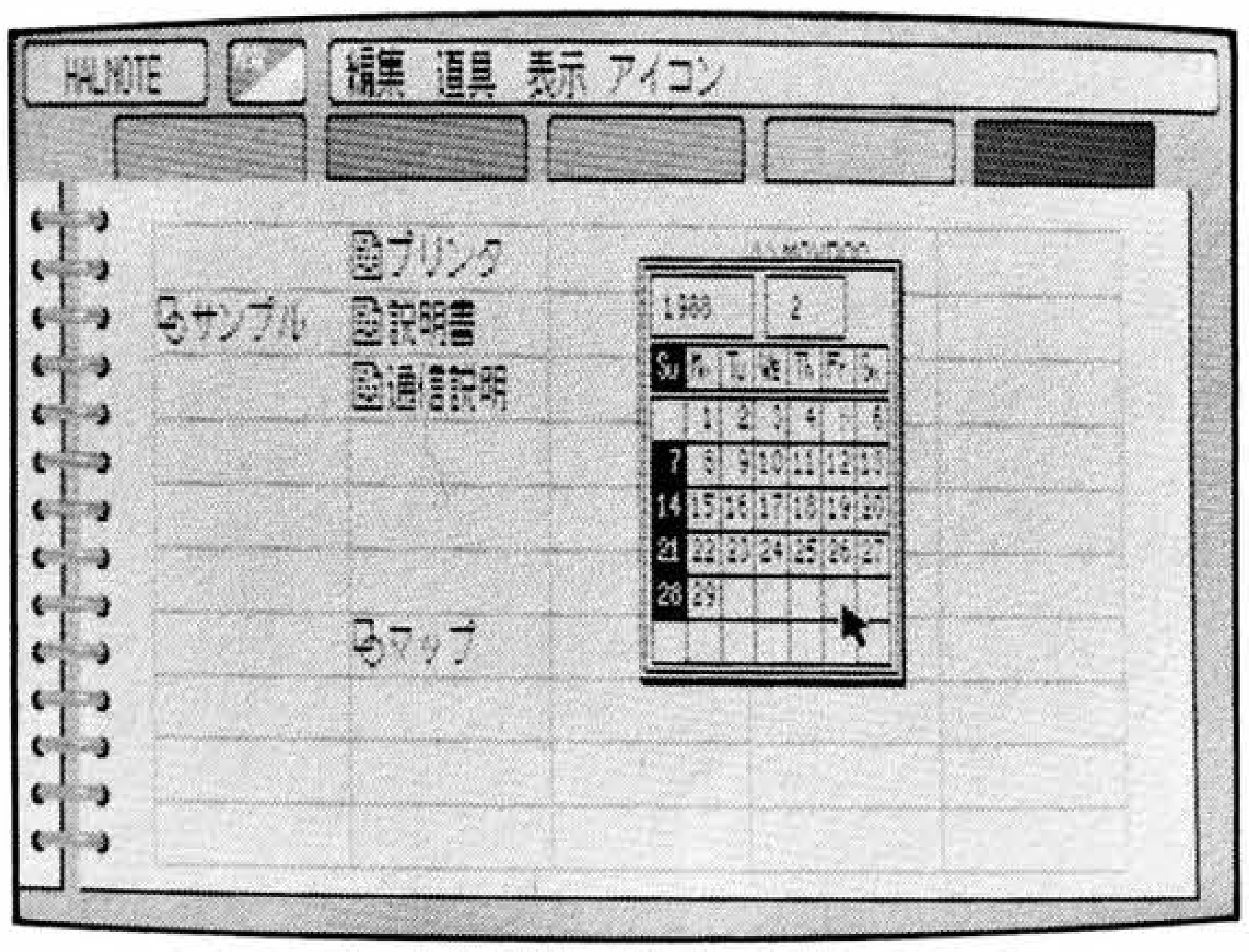


写真 6.13 カレンダの呼び出し

写真 6.13 は，カレンダを呼び出したところです．デスクアクセサリーは，バインダからだけではなく，ワープロや図形プロセッサを使っている最中でも，自由に呼び出すことができます．いつでも使うという機能ではありませんが，使い込んでいるとこんな機能もうれしいものです．

HALNOTE 上のソフトウェア

HALNOTE には非常に強力な日本語ワードプロセッサ(以下ワープロと呼ぶ)と図形プロセッサが含まれています．とくに，ワープロについては，さまざまなフォントの文字をさまざまな大きさで自由に扱え，影を付けたり，斜体文字にするなどの文字装飾も自由自在に行える強力なものとなっています(**写真 6.14**)．

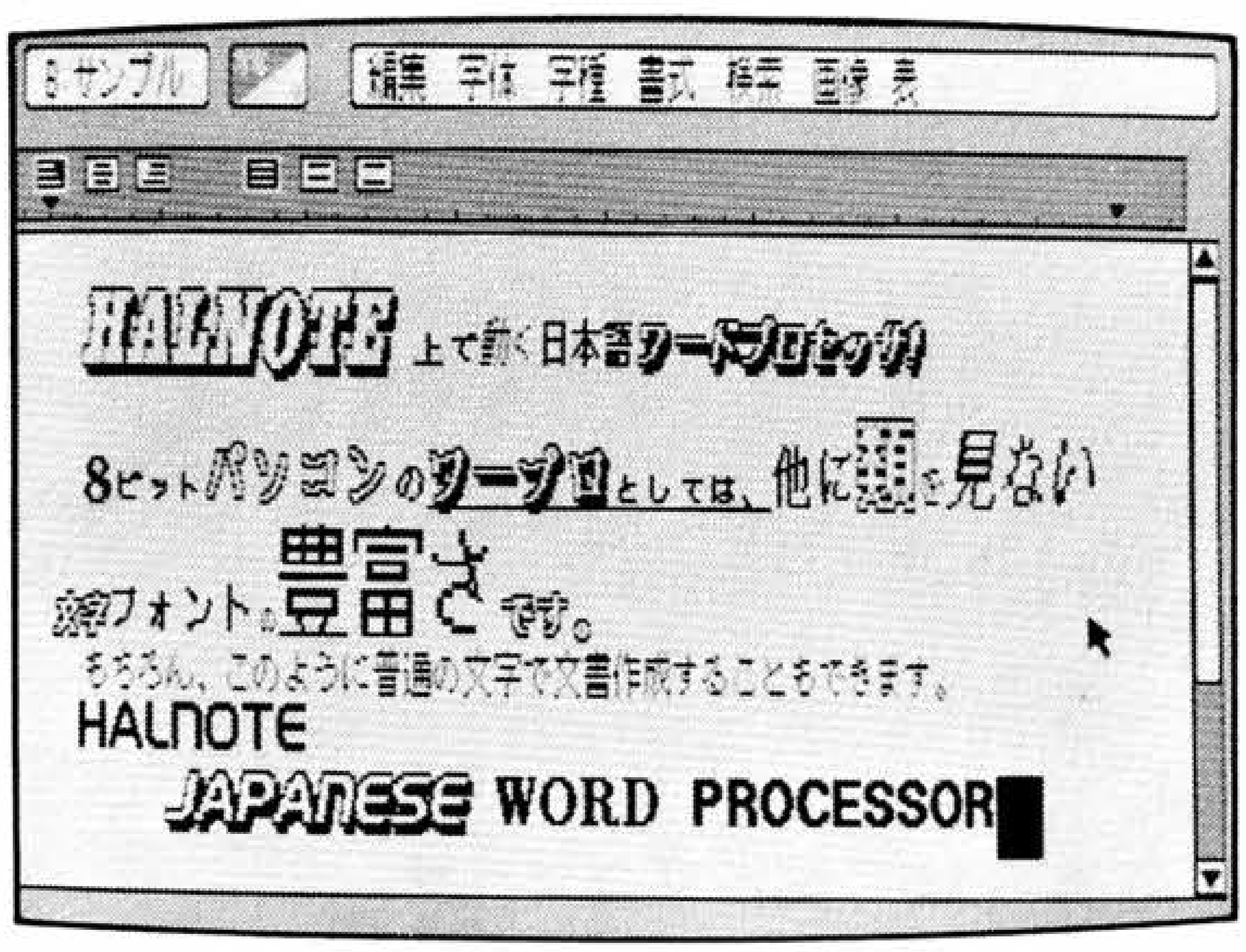


写真 6.14 さまざまなフォントを持つ日本語ワードプロセッサ

文字の種類の豊富さという点では，ここまで強力なワープロは8ビット・マシンでは，ほかに例がありませんし，16ビット・マシンでもめずらしいでしょう．

また，漢字変換機能も非常に強力で，16ビット・マシン上のフロントエンド・プロセッサ「ATOK」とほとんど同じ操作感覚で扱えます．このため，ワープロ「一太郎」などを使ったことのある人ならば，何も予備知識なしにこのワープロを使うことができます．

このワープロで作った文書は，HAL標準形式とMSX-DOS形式の2通りのファイル形式で保存しておくことができます．HAL標準形式では，フォント，画像，書式などのデータを含んだ形で記録されるために，MSX-DOS上の他のアプリケーション・ソフトウェア(MSX-Writeなど)を使って普通に読むことはできません．

一方，MSX-DOS形式では，あらゆる文字属性を無視して文字データのみを記録するために，この方式で記録されたテキスト・ファイルは，MSX-Writeや，さきほど作ったKTYPEなどで読むことができます．逆に，MSX-Writeで作ったファイルをHALNOTEで読み込んで，文字を装飾することも可能です．

次に，図形プロセッサについてですが，これもマウスを使ってさまざまな図形を描くことができます．もちろん，さまざまな文字フォントや強力な漢字変換機能はここでも使えますので，製図など文字と図形を組み合わせて使いたい場合には便利でしょう(**写真6.15**)．



写真6.15　図形プロセッサ

以上のようにHALNOTEはMSX-DOSを視覚的にとらえ，マウスを使った直接的な指定方法でコマンドを与えることができます.

現在のところ，HALNOTE上でのアプリケーション・ソフトウェアは，ワープロと図形プロセッサ，今回紹介できなかった通信プロセッサの3つだけですが，これから他のアプリケーション・ソフトウェアも発売される予定です.

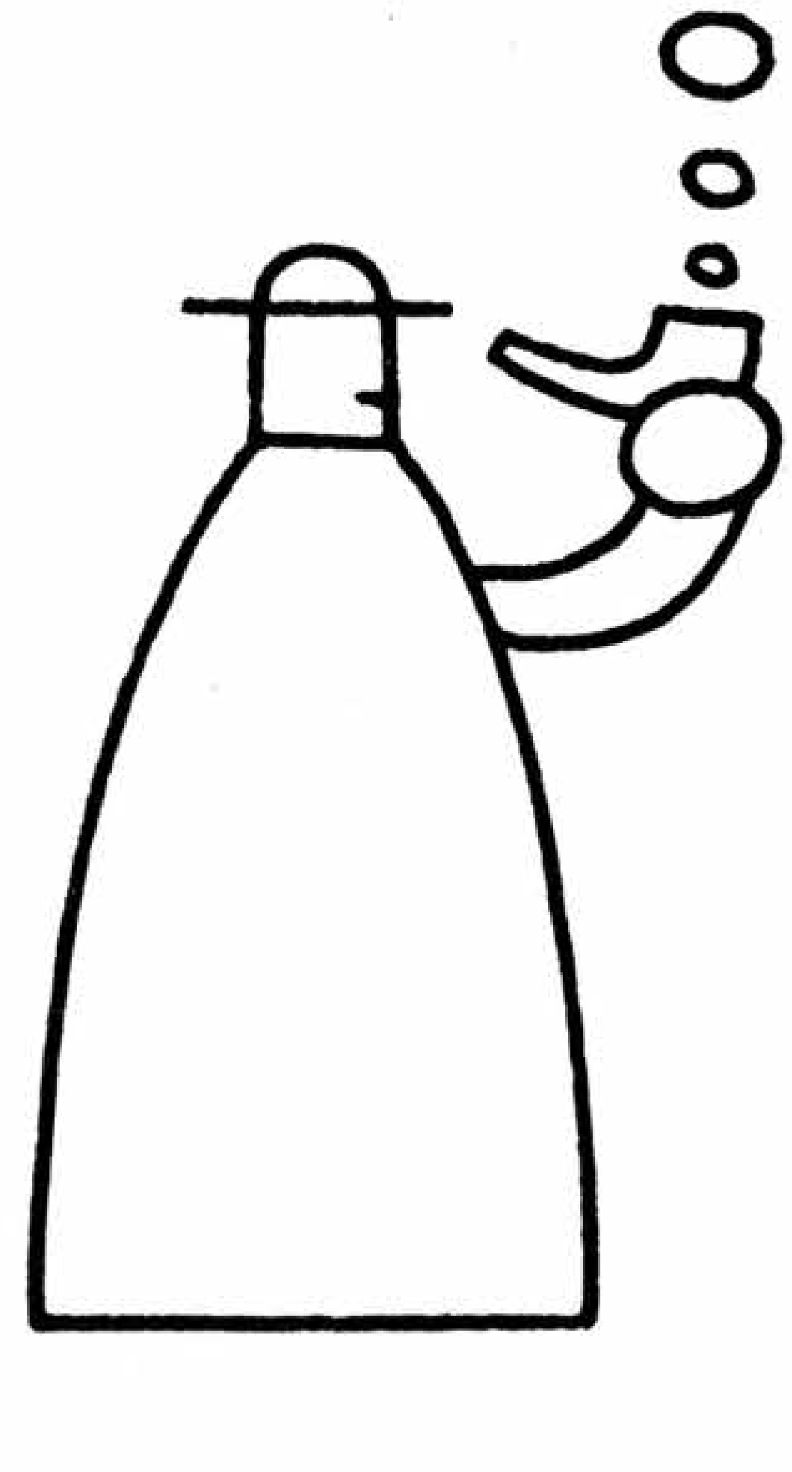
MSX-DOSが，単にプログラムの開発環境として使用されるのではなく，使いやすいアプリケーション・ソフトウェアの土台となるために，HALNOTEは大いに貢献するものと思われます.

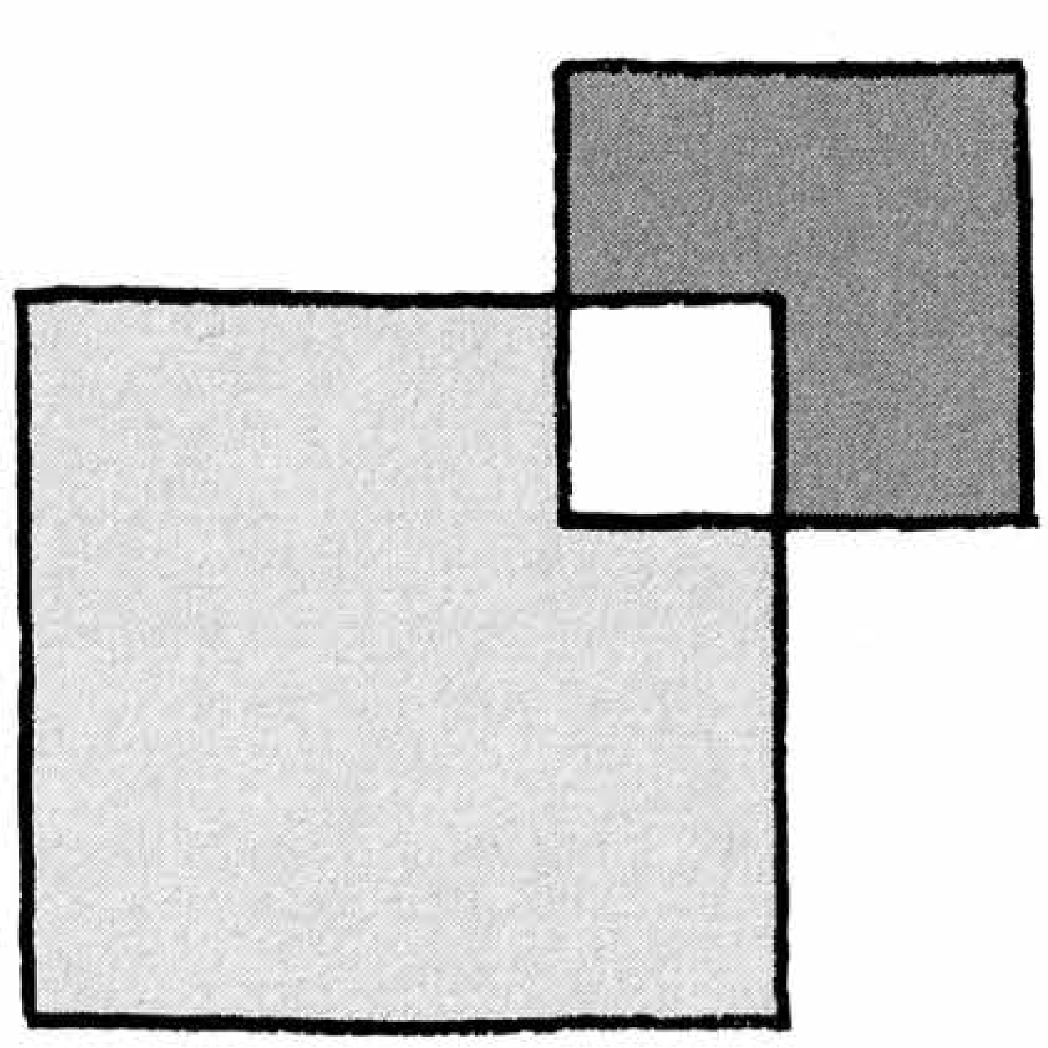


7章

プログラム開発の手引

— アセンブリ言語とC言語による
プログラミング —





7-1 プログラム開発ツール

MSX-DOS 上でのプログラム開発は，いったいどのようになっているのでしょうか？　本章では，MSX-DOS 上に提供されている多くの言語の中から代表的なものを選び，それらを利用した実際のプログラム開発手順を追うことにより，MSX-DOS のプログラム開発環境を見ていきたいと思います．

プログラム開発と言語

プログラミングといえば，まず思い浮かぶのは BASIC です．もし，DISK-BASIC 上でプログラムを作るとしたら，言語は BASIC を使うしかありません．それに比べて，MSX-DOS 上ならば豊富に揃った言語の中から，開発するプログラムに合わせて最も適したものを使用することができます．

また，DOS 上で豊富に揃っているのは，言語だけではありません．プログラミングのために役立つ開発ツールもさまざまなものが用意されています．たとえば，BASIC 上でプログラムを入力するとしたら，BASIC に備わっているスクリーン・エディタ以外は使えません．しかし，DOS 上なら自分の好みに合ったエディタを選んで，プログラムの入力することができます．

さて，MSX-DOS 上でプログラムを開発するということは，最終的にはコマンド・レベルから直接実行可能な COM ファイルを作り出すことを意味します．しかし，COM ファイルはマシン語です．本来，コンピュータが理解実行できる言語はマシン語だけであり，どのような言語を用いても最終的に実行しているときはマシン語となっています．したがって，コンピュータの性能を限界まで引き出すには，プログラムをあらかじめマシン語で用意しておくことが必要です．ところがマシン語のプログラムは 16 進数，さらには 2 進数の羅列で表されます．これでは，普通の人間には，とても理解することができません．そのため，マシン語の 1 語 1 語を人間にとって意味のある記号

に置き換えた**アセンブリ言語**や，マシン語をいくつも組み合わせて，より人間の言葉に近い表現に置き換えた**高級言語**などが広く使われています．これらの言語で書かれたプログラムをソース・プログラムと呼びます．そして，そのソース・プログラムを，コンピュータを使ってマシン語プログラムに変換していきます．このソース・プログラムをマシン語に変換する作業を**コンパイル**（高級言語の場合），**アセンブル**（アセンブリ言語の場合）と呼び，これらの作業を行うプログラムのことをそれぞれ**コンパイラ**，**アセンブラ**といいます．

アセンブリ言語は高級言語ほどわかりやすくはありませんが，マシン語と1対1で対応しており，マシン語でできることはすべて記述できるため，こまかい処理が可能です．このように，高級言語とアセンブリ言語にはそれぞれ一長一短があり，どちらがよいとは一概にいうことはできません．MSX-DOS上にはさまざまなコンパイラや本格的なアセンブラがありますが，ここではアセンブラを例にして，プログラムを開発する手順について説明することにしましょう．

プログラム開発の手順

図7.1はアセンブリ言語を使ってプログラムを作る手順を示したものです．そのほかの高級言語を使ってプログラムを作る場合もだいたい同じ手順です．

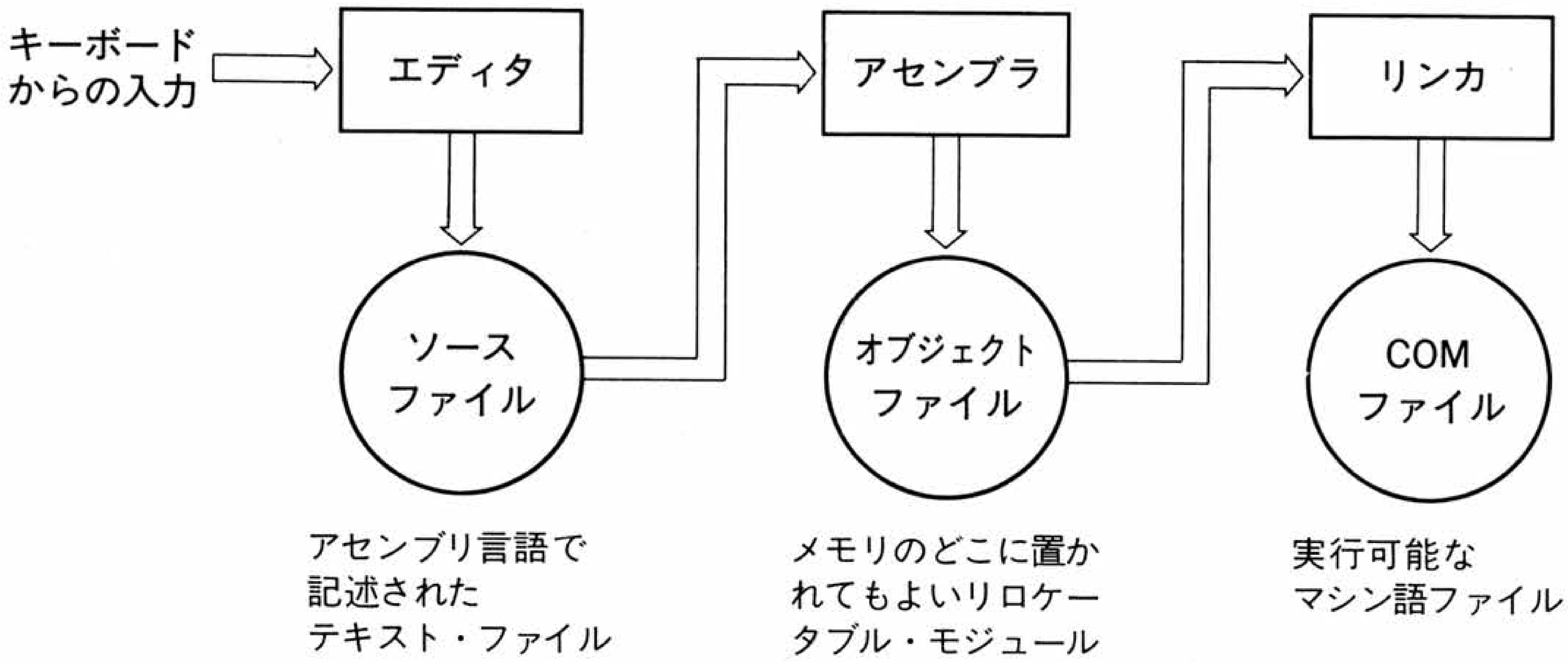


図7.1　アセンブリ言語からCOMファイルのできるまで

プログラムの構想を練り，紙上での検討が十分にすんだとしましょう．そこで，まず行うのはアセンブリ言語で記述されたソース・プログラムをエディタで入力することです．ソース・プログラム自体は普通のテキスト・ファイルですから，デバイス・ファイルCONを使ったり，BASICから入力することも可能ですが少し大きいプログラムになるとそれでは効率が悪くて，とても実用にはなりません．やはり，エディタの豊富な編集機能は必要です．たとえば，MSX-DOS上のエディタとしては「MSX-DOS TOOLS」に含まれているMEDがあります．

ソース・プログラムが入力できたら，それをアセンブルしなくてはなりません．MSX-DOSで使用できるアセンブラはいくつかありますが，ここではM80というアセンブラを使ってみることにしましょう．このM80は，豊富な機能を持っており，CP／M上でも使われている本格的なアセンブラです．

ところで，DOS上でプログラムの開発を行う際には，〝**モジュール別プログラミング**〟と呼ばれる手法がとられます．これは，プログラムを，モジュールと呼ぶいくつかのかたまりに分けて開発し，最後にそれらをつなぎ合わせて１つのプログラムにするプログラム作成法です．各モジュールを独立させることにより，さまざまな利点があります．たとえば，プログラムの設計をしやすい，仕様さえ決まれば複数の人間でモジュールごとに開発できる，プログラムの仕様を変更したい場合もモジュールの交換ですむ，などがあげられます．

また，モジュール別に開発する利点はこれだけではありません．あるプログラムのために作ったモジュールであっても，他のプログラムで同じような処理が必要なら，このモジュールを流用することができるのです．さらに，モジュールをどのような言語で開発してもかまいません．つまり，アセンブリ言語で開発したモジュールと高級言語で開発したモジュールを組み合わせて使うことも可能なのです．プログラムを何本か作って，汎用性の高いモジュールが蓄積されると，これらのモジュールをつなぎ合わせることによってある程度のプログラムを作ることが可能となり，プログラミングの労力が大いに軽減されます．

さて，話をアセンブラにもどしますが，アセンブラはソース・プログラムを解釈してマシン語に直していき，**オブジェクト・ファイル**と呼ばれるマシン語

のファイルを生成します．いま，モジュール別プログラミングの利点の１つとして述べたように，このオブジェクト・ファイルは，さまざまなプログラムと組み合わせて使われることを考えておかねばなりません．そのため，このオブジェクト・ファイルはメモリ上のどこに置かれてもかまわない形式になっており，**リロケータブル(再配置可能)・モジュール**などと呼ばれることもあります(**図7.2**)．

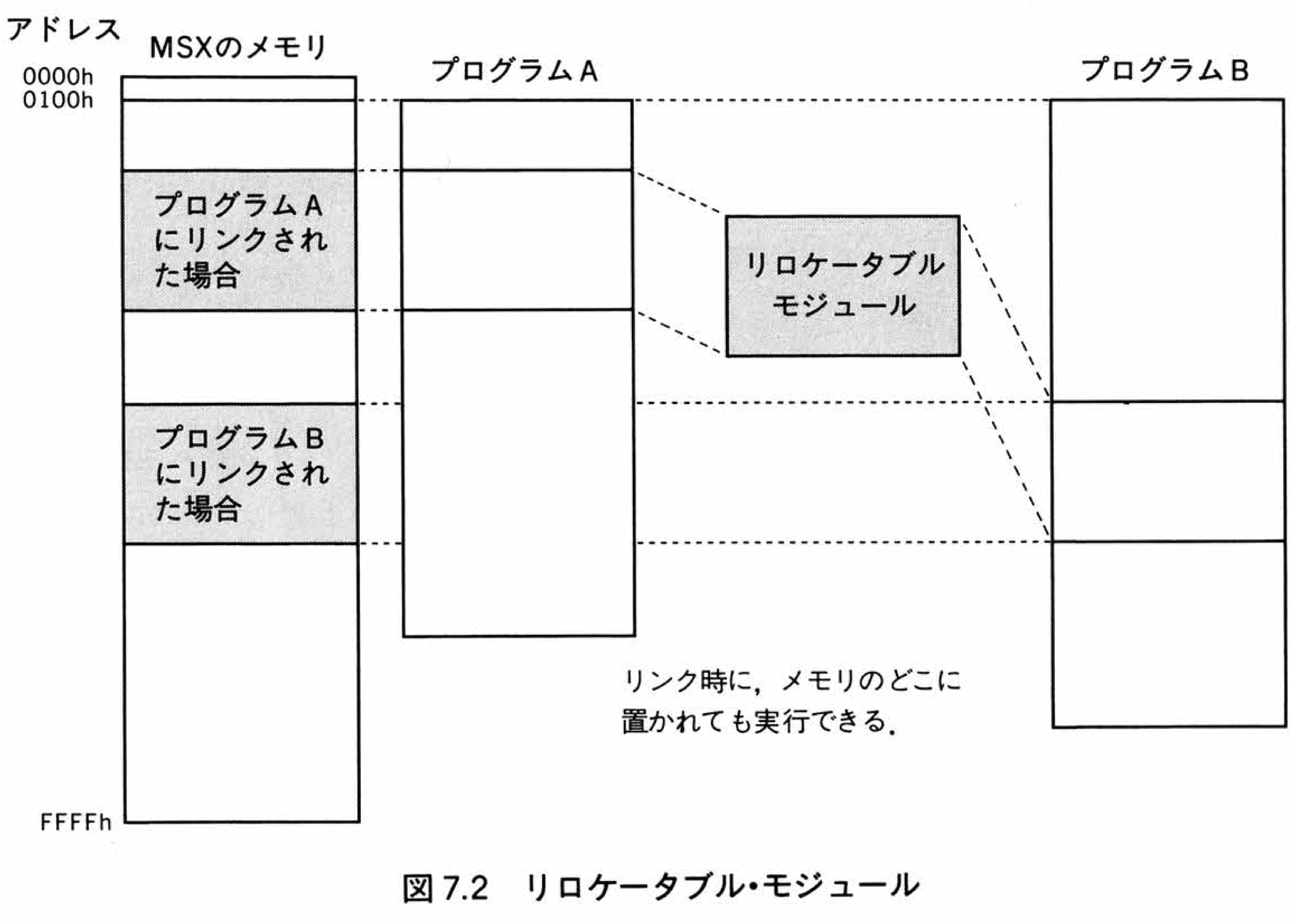
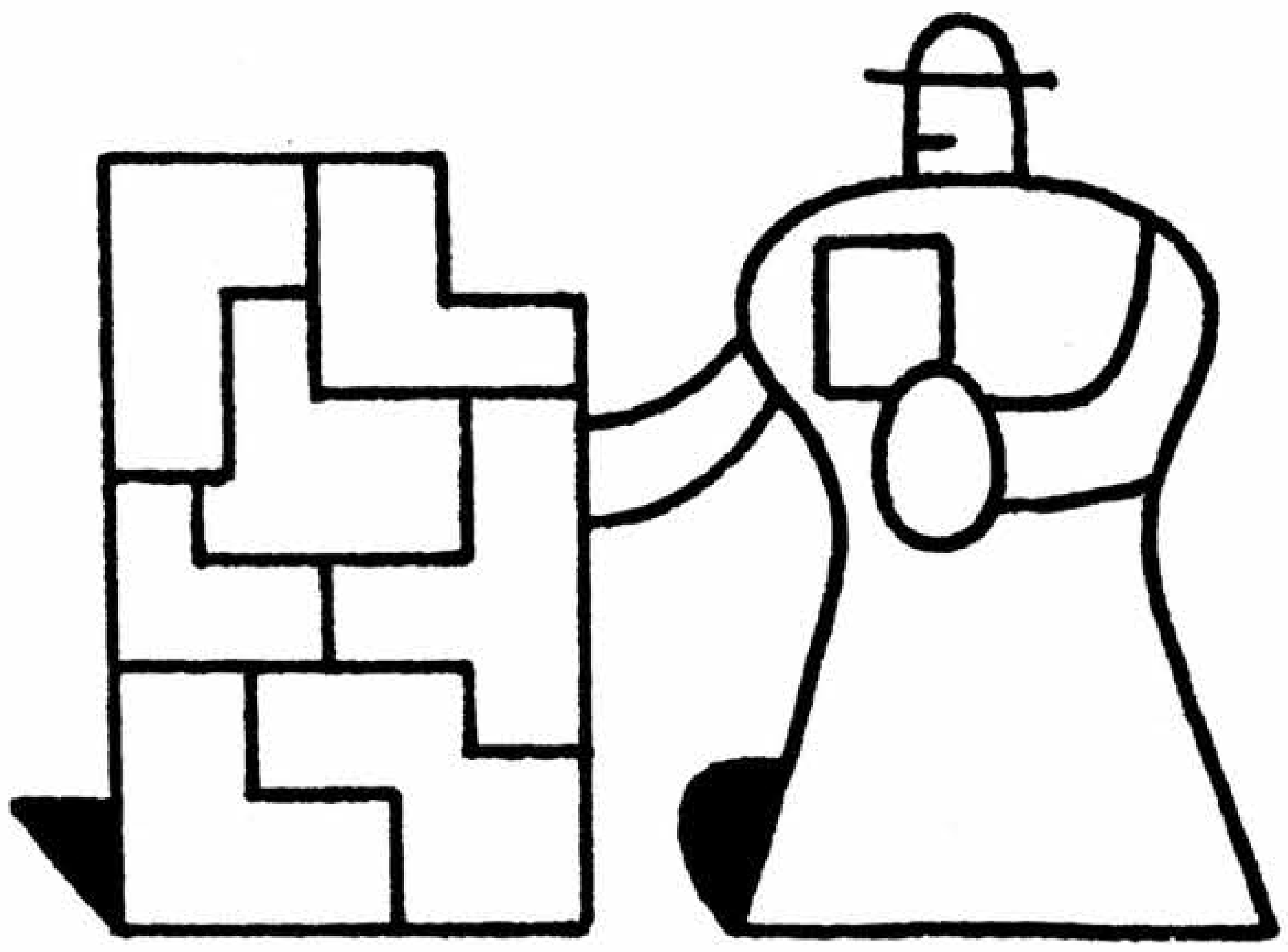


図7.2　リロケータブル・モジュール

オブジェクト・ファイルはメモリ上のどこにでも配置できる形式のため，プログラムの分岐先や変数の格納場所の，具体的なメモリ上での位置(アドレス)がまだ決まっていません．この段階では，アドレスの代わりにラベルが付けられています．そのため，このままではプログラムとして実行することはできません．

もし，ここまでで重大なエラーを生じなければ，このオブジェクトファイルから実行可能なCOMファイルを作ります．この作業は**リンカ**によって行われます．MSX-DOSのリンカには，L80と呼ばれるものを使います．



リンカは，いくつかのオブジェクト・ファイル(1つでもかまわない)をつなげ，ラベルに実際のアドレスを割り当てます．こうしてリンカが出力するCOMファイルはコマンド・レベルから即実行可能なプログラムとなります．

しかし，プログラムは完璧を期して作ってもバグ(プログラムの誤り)を生じるものです．普通はソース・プログラムを見ながらデバッグ(バグを取り除く作業)を行いますが，中にはソース・プログラムをながめただけではわからないバグもあります．こんなときには，デバッガと呼ばれるプログラムを使って，オブジェクト・ファイルを直接メモリに読み込んで，プログラムの一部を実行させてはストップさせ，レジスタやメモリの内容を直接見ながらバグを探したりします．そして，バグが見つかったらソース・プログラムを修正して再びアセンブル，リンクを行うわけです．このように，プログラムの完成までには，アセンブル，リンクという処理は何度も行います．バッチ・ファイルをうまく使えば，この手間を軽減することができます．

MSX-DOSには，「**MSX-S BUG**」(株式会社アスキー)と呼ばれるシンボリックデバッガ(ラベルなどを直接取り扱える強力なデバッガ)があり，しつこいバグ退治には欠かせないツールの1つとなっています．

また，6章で紹介したようにMEDやM80，L80は「MSX-DOS TOOLS」に含まれています．高級言語を使う場合でも，アセンブリ言語のソース・プログラムやオブジェクト・ファイルを生成するものについては，M80やL80がないとCOMファイルが作れません．MSX-DOSを本格的に使うには，「MSX-DOS TOOLS」は必需品といっても過言ではありません．

7-2 システムコールの役割

1章でも述べましたが，DOSにキーボードからの入力やディスプレイへの出力，ディスクのアクセスなどの基本的なルーチン(決まった仕事をするプログラム)が組み込まれているので，これらのルーチンを利用することによって，プログラミングを効率に行うことができます．この点に関して，ここでもう少し具体的な話をしてみましょう．**図7.3**を見てください．いま，Aさ

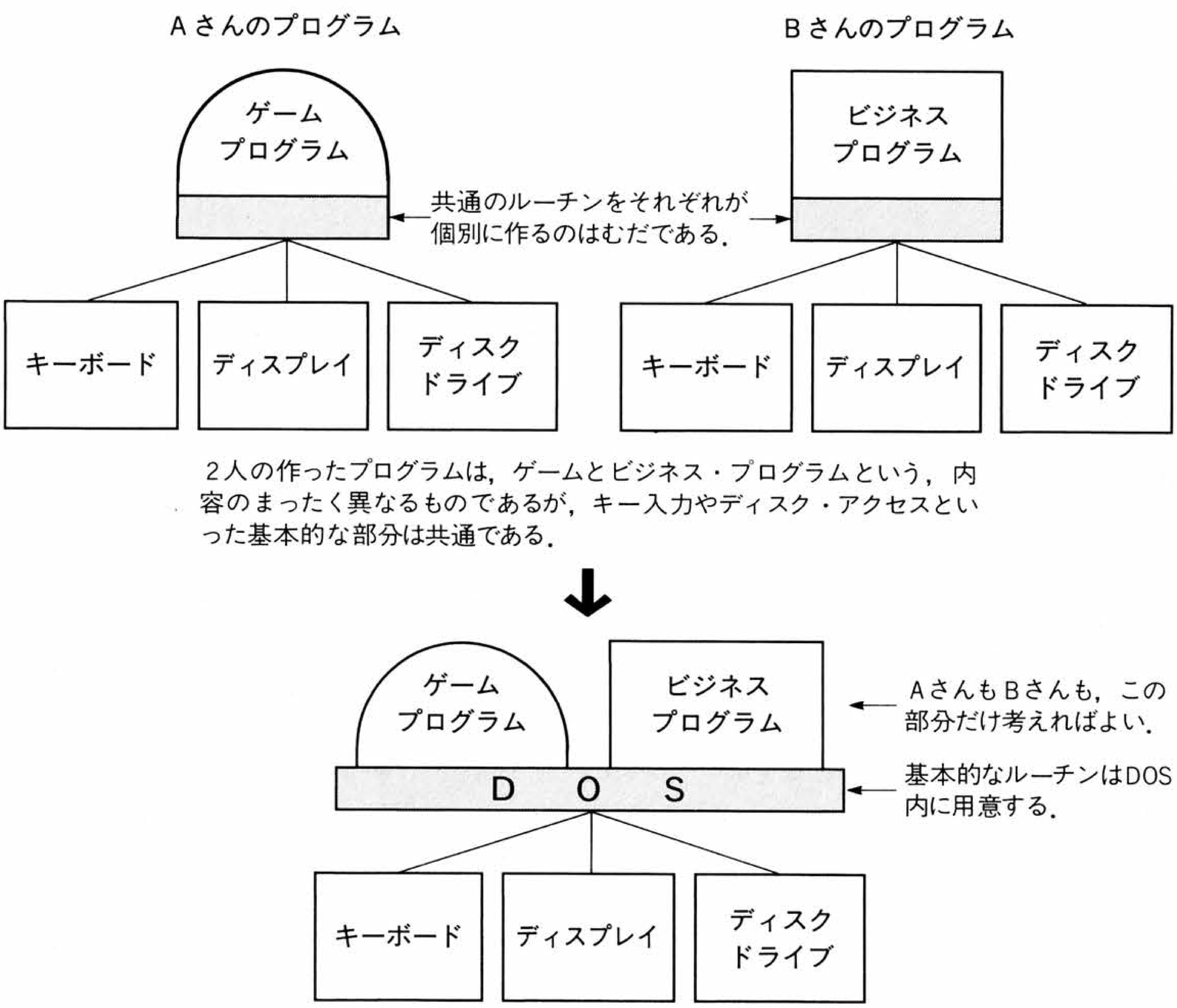


図7.3 システムコールの考え方

んがゲームの，Bさんがビジネス用のプログラムを作っているとします．各自，プログラムのすべての部分を自作したとします．当然，2人のプログラムはまったく違うものになります．しかし，よく見てみると，Aさんのプログラムでキーを読み込むルーチンや得点を表示するルーチンは，Bさんのプログラムでデータを入力したり，売り上げを表示するルーチンと同じものが使えるはずです．もし，あらかじめ，キーボードから文字を読み込んだり，画面に文字を表示したりする汎用的なルーチンが用意されていたら，2人の作業はより軽減されるのではないでしょうか．

誰もが使うことのできる，基本的で汎用的なルーチンの提供．この考え方からDOS内に用意されている基本ルーチンが**システムコール**です．DOS上のソフトはこのシステムコールを使うことによってプログラムの簡略化をはかることができます．

プログラムの簡略化だけがシステムコールの利点ではありません．システムコールを使うことでプログラムの移植性や互換性を高めることができます．

マシンによって，VRAM(画面のデータを記憶しておくメモリ)のアドレスは違っているのが普通です(MSXどうしならきちんと統一されているが，他のCP／Mマシンなどとは違う)．このため，マシンが変われば，同じプログラムでは画面に文字を表示することすらできないことになります．この場合でも，システムコールの中身は，それぞれのマシンでDOSが管理してくれるので，ユーザーがこれを気にすることはありません．つまり，文字出力のシステムコールを使っているプログラムはマシンが変わっても，同じDOSを使っている限りきちんと文字を表示してくれるのです．

とくにMSX-DOSのシステムコールはCP／Mと共通のものが多いので，CP／Mの豊富なソフトを容易に移植することが可能です．

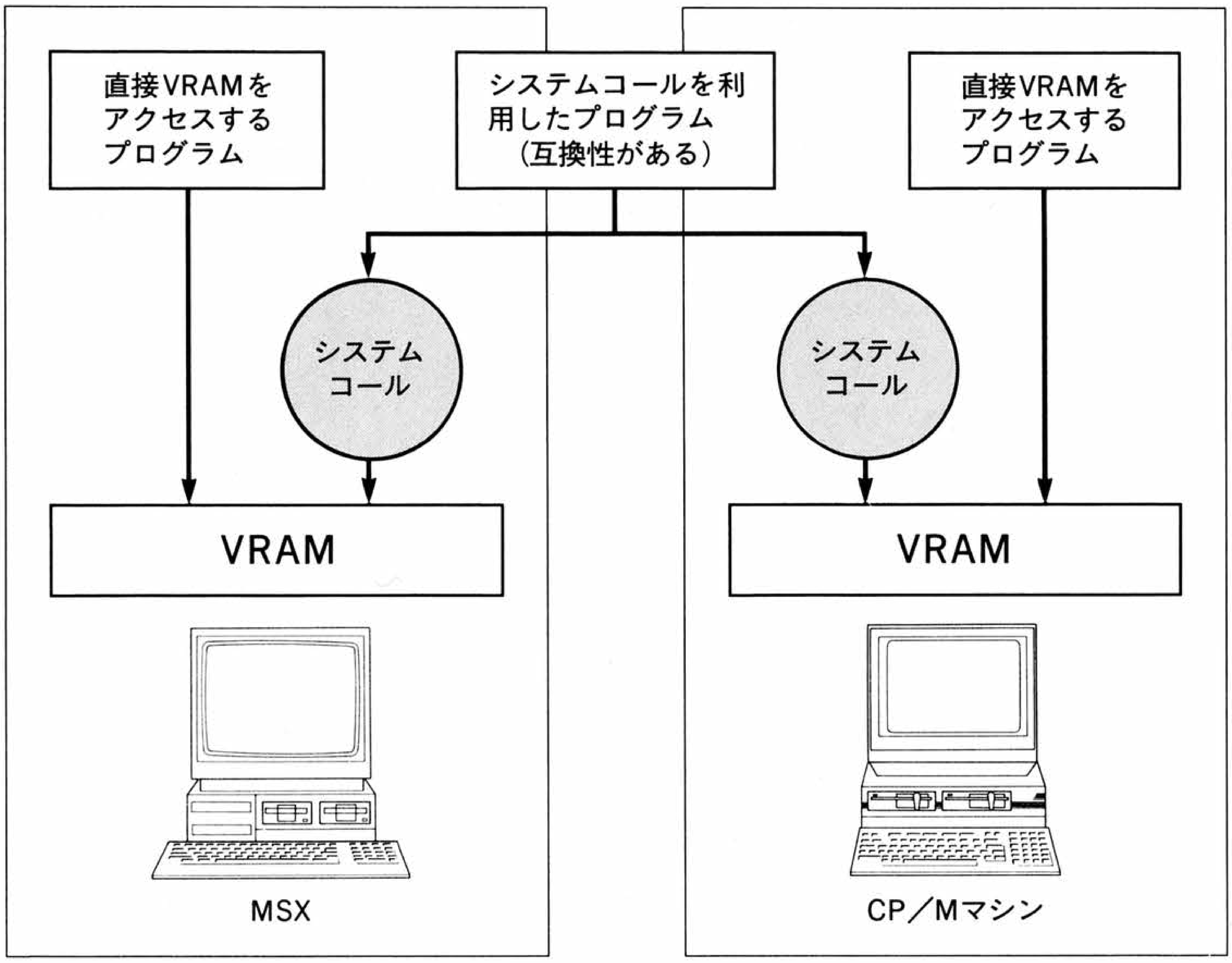


図 7.4　システムコールによるCP/Mマシンとの関係

MSX-DOSのシステムコールには**表 7.1**のようなものがあります．

これらのシステムコールはプログラム中で，ファンクション・コールの番号を決められた手続きで指定するだけで使うことができます．ただし，実際に使用するには，マシン語の知識のほかにも，CPUの構造やMSX-DOSの内部構造を知る必要があります．「MSXマシン語入門講座」，「MSX2テクニカル・ハンドブック」，「MSX-DOSスーパー・ハンドブック」(いずれも株式会社アスキー)などを読まれたうえで，マシン語プログラムに挑戦されることをおすすめします．

マシン語を使わない人ならシステムコールを使う機会はあまりないかもしれません．しかし，高級言語でもCなどを使えば，システムコールを使うこともありえますから，まったく関係ないと簡単に片付けてしまうわけにはいきません．

やはり，DOS を語るにシステムコールを欠くことはできません．縁の下の力持ち的存在ですが，システムコールというものがあるということは覚えておいてください．

ファンクションNo.	機　能
00H	システムリセット
01H	コンソールから1文字入力(入力待ちあり，エコーバックあり，コントロール・コードチェックあり)
02H	コンソールへ1文字出力
03H	補助入力装置から1文字入力
04H	補助出力装置へ1文字出力
05H	プリンタへ1文字出力
06H	コンソールから1文字入力(入力待ちなし，エコーバックなし，コントロール・コード・チェックなし)／1文字出力
07H	コンソールから1文字入力(入力待ちあり，エコーバックなし，コントロール・コード・チェックなし)
08H	コンソールから1文字入力(入力待ちあり，エコーバックなし，コントロール・コード・チェックあり)
09H	文字列出力
0AH	文字列入力
0BH	コンソールからの入力チェック
0CH	バージョン番号の獲得
0DH	ディスク・リセット
0EH	デフォルト・ドライブの選択
0FH	ファイルのオープン
10H	ファイルのクローズ
11H	ワイルドカードに一致する最初のファイルの検索
12H	ワイルドカードに一致する2番目以降のファイルの検索
13H	ファイルの抹消
14H	シーケンシャルなファイルの読み出し
15H	シーケンシャルなファイルの書き出し
16H	ファイルの作成
17H	ファイル名の変更
18H	ログイン・ベクトルの獲得
19H	デフォルト・ドライブ名の獲得
1AH	DMAアドレスの設定
1BH	ディスク情報の獲得
1CH～20H	無効
21H	ランダムなファイルの読み出し
22H	ランダムなファイルの書き込み
23H	ファイル・サイズの獲得
24H	ランダムレコード・フィールドの設定
25H	無効
26H	ランダム・ブロック書き込み
27H	ランダム・ブロック読み出し
28H	ランダムなファイルの書き込み(不用部を00Hで埋める)
29H	無効
2AH	日付の獲得
2BH	日付の設定
2CH	時刻の獲得
2DH	時刻の設定
2EH	ベリファイ・フラグの設定
2FH	論理セクタの読み出し
30H	論理セクタの書き込み

表 7.1　システムコール一覧表

7-3 アセンブラによるプログラミング

では，ここでMSX-DOS上でのプログラムの開発のサンプルとして，アセンブラを使って何かプログラムを作ってみましょう．

本節の作業を実際に行うには「MSX-DOS TOOLS」が必要ですが，文章を追って行けば雰囲気はつかめると思います．

ここでは，文字列を表示するコマンドであるECHOを作ってみましょう．プログラムを作るには，まずそのプログラムが何をしたらいいのかを十分に検討して仕様を決めなくてはなりません．プログラムが文字列を表示するという単純なものなので，紙上での検討は十分にしたことにしましょう．そしてプログラムも**リスト7.1**のようにできあがったことにします．

```
        DMA     EQU 0080H
        SYSTEM  EQU 0005H

        .Z80

        PUSH    AF
        PUSH    BC
        PUSH    HL
        PUSH    IX

        LD      HL,DMA
        LD      IX,BUF
        LD      C,(HL)
        LD      B,0FFH   ;FLAG = OFF
        INC     HL
        INC     C

LOOP:   DEC     C
        JR      Z,BREAK
        LD      A,(HL)
        CP      '$'
        JR      NZ,NESC
```

```
        LD      D,B      ;IF (B == OFF) GOTO NESC
        INC     D
        JR      Z,NESC
        LD      A,01BH

NESC:   CP      022H     ;'"'
        JR      NZ,NORMAL

        LD      A,B
        CPL              ;REVERSE FLAG
        LD      B,A
        JR      SKIP

NORMAL: LD      D,A
        LD      A,B
        AND     A        ;IF (B == ON) GOTO COPY
        JR      Z,COPY

        LD      A,D      ;IF (A = ' ') GOTO SKIP
        CP      ' '
        JR      Z,SKIP

COPY:   LD      (IX),D
        INC     IX

SKIP:   INC     HL
        JR      LOOP

BREAK:  LD      (IX),'$'
        LD      A,B
        AND     A
        JR      Z,ERROR
        LD      DE,BUF
EXIT:   LD      C,09H    ;PRINT STRING ………システムコールを利用している.
        CALL    SYSTEM
        POP     IX
        POP     HL
        POP     BC
        POP     AF
        RET

ERROR:  LD      DE,MESSEG
        JR      EXIT

BUF:    DS      256
MESSEG: DB      'syntax error$'

        END
```

リスト 7.1　ECHO.ASM

では，エディタを使ってプログラムの入力をするところから解説を初めます．**図 7.5** はエディタ MED を使ってソース・プログラムを入力しているところです．いま，2 ドライブの MSX を使い，ドライブ B に TOOLS のディスクがはいっているとします．

```
DMA      EQU 0080H
SYSTEM   EQU 0005H

.Z80

PUSH     AF
PUSH     BC
PUSH     HL
PUSH     IX

LD       HL,DMA
LD       IX,BUF
LD       C,(HL)
LD       B,0FFH   ;FLAG = OFF
INC      HL█

+-------+-------+-$-----+-------+-------+-------+-------+-------+-------+-------
1       9       17      25      33      41      49      57      65      73
```

MSX-DOS TOOLSに含まれているエディタ〝MED″を使って，アセンブリ言語のソース・プログラムを入力していく．
このとき，ファイル名はECHO. ASMにする．

図 7.5　エディタでの入力

プログラムを間違いなく入力できたら，ECHO.ASM というファイル名でセーブします．次はアセンブルです．**図 7.6** のように，M80 を起動してアセンブルを行います．

```
A>b:m80 =echo.asm ⏎ ……………アセンブラM80の起動.

No Fatal error(s) ……………エラーもなく，無事アセンブル終了.

A>
```

図 7.6　アセンブル

もし,エラーを生じたらエラーメッセージを頼りにしてソース・プログラムの誤りを直します. アセンブルの最中に生じるエラーは, 命令のスペルの打ち間違えなどの単純なミスが多く比較的対処しやすいものです.

無事, エラーも出ずアセンブルがすめば, **図7.7** のようにL80を使ってリンクを行います.

```
A>b:l80 echo,echo/n/e         ……………リンカL80の起動.

MSX.L-80  1.00  01-Apr-85    (C)  1981, 1985 Microsoft

Data      0103     0265     <  354>

43224 Bytes Free
[0000     0265          2]    ……………無事終了.

A>dir echo.*
ECHO     ASM     1440 87-12-23   3:13p   ……アセンブラのソース・ファイル
ECHO     REL      256 87-12-23   3:13p   ……オブジェクト・ファイル
ECHO     COM      384 87-12-23   3:17p   ……COMファイル
        3 files    600064 bytes free
A>
```

図7.7 リンク

L80では, スイッチを指定することによりさまざまな機能を実現することができます. しかし, これらのスイッチは複雑なので, ここでは説明を省略します. 上に示したように指定すれば, リンクが終了したあと, ディスク上にECHO.COMを作ってくれます.

リンクも無事終了したら, 動作チェックを行います. もし, 動作が予定と違う場合は少しやっかいです. ソース・プログラムをながめて原因がすぐにわかればよいのですが, バグというのはそんなに甘いものではありません. アセンブリ言語はわかりやすいとはいってもマシン語を直接扱うのと比べればの話で, 決して苦労なしに読めるというものではありません. ソース・プログラムを見て原因がわからなければ, 「MSX-S BUG」などのデバッガを使って

動作チェックを行いながらバグを取り除いていくことになります．実際のプログラミングでは，プログラムを作るよりも，このデバッグ作業の方が大変なことが多いようです．やはり，プログラムを作る際には，あらかじめ検討に次ぐ検討を重ね，できる限りバグを出さないようにするのが効率のよいプログラミングということになります．

さて，こうしてデバッグが終われば，いよいよ完成です(図 7.8)．

```
A>echo "msx dos!" ! ! ! ! ⏎
msx dos!!!!!
A>echo "m$Bs$Bx$B-$Bd$Bo$Bs" ⏎ ……………"$B"がESCシーケンス
m
 s
  x
   -
    d
     o
      s
A>
```

図 7.8　ECHO コマンドの実行例

ここで作った ECHO コマンドは，TOOLS に含まれる ECHO と少し仕様が違うので，使い方を説明しておきましょう．表示する文字列をダブルクォーテーション(")で囲むと文字列に含まれるスペースも表示します．ただし，ドル記号($)は，ファンクション・コール 09h 番のエンドマークですので表示できません．しかし，ダブルクォーテーションで囲んだ中身で $ を使うと，ESC(1Bh)として認識されます．

この ESC というのは EOF(p.78を参照)などと同じように特別な意味を持つ文字です．ESC と次にくる数文字の組み合わせによってカーソルの移動とか，画面の消去などのコントロール文字を表現することができます．

このように，ESC＋数文字で表される特別な文字を **ESC シーケンス(エスケープ・シーケンス)** といいます．次に ESC シーケンスの一覧表をあげておきます(ESC シーケンスでは英文字の大文字，小文字を区別しますので注意

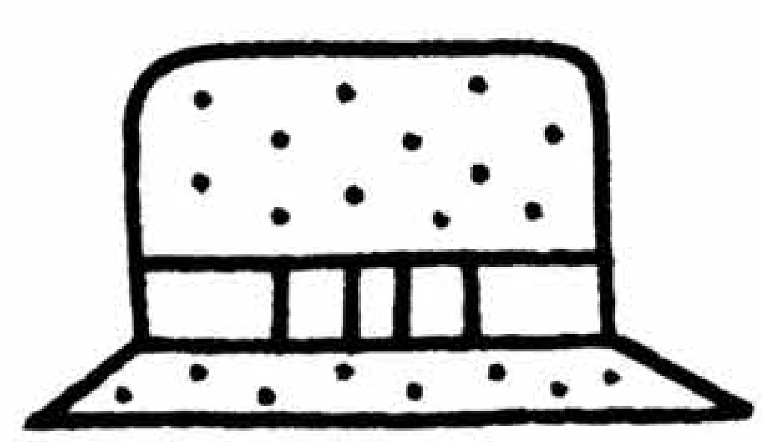
してください).

これらのESCシーケンスをECHOコマンドでいろいろ試してみてください. 5章のバッチ・ファイルと組み合わせるのも面白いかもしれません. ECHOコマンドを使ってカーソルを画面上の好きな場所に移動させてから, メッセージを表示させるなどの応用ができます.

○カーソル移動	
<ESC>A	カーソルを上に移動
<ESC>B	カーソルを下に移動
<ESC>C	カーソルを右に移動
<ESC>D	カーソルを左に移動
<ESC>H	カーソルをホームポジションに移動
<ESC>Y<Y座標+20H><X座標+20H>	カーソルを(X,Y)の位置に移動
○編集, 削除	
<ESC>j	画面をクリア
<ESC>E	画面をクリア
<ESC>K	行の終わりまで削除
<ESC>J	画面の終わりまで削除
<ESC>L	1行挿入
<ESC>M	1行削除
○その他	
<ESC>x4	カーソルの形を'■'にする
<ESC>x5	カーソルを消す
<ESC>y4	カーソルの形を'_'にする
<ESC>y5	カーソルを表示する

表 7.2　ESC シーケンス一覧表

ESCシーケンスはDOS上でカーソルを制御するためには必要な知識です. ぜひ, ここで使いこなせるようにしておいてください.



7-4 高級言語によるプログラミング

最後に高級言語を使ったプログラミングを紹介しましょう．高級言語にもFORTRANやPASCAL，LISP，果てはBASICまでさまざまなものがありますが，ここでは，高級言語の中でも比較的アセンブラに近く，きめのこまかいプログラミングが可能だといわれるC言語を使ってみましょう．

C言語はもともと，米国のベル研究所でUNIXというOSを開発するために作られた言語で，DOSと密着したプログラムを書くには最適ともいわれています．

MSX-DOS上には，「**MSX-C**」というCコンパイラが用意されています．このCコンパイラはアセンブラのソース・プログラムを生成してくれるので，このソース・リストを見てさらに自分で効率よいプログラムにしてからアセンブルすることも可能です．

では，C言語のサンプル・プログラムとして，ファイルの内容を比較する（ファイル・コンペア）コマンドFCを，MSX-Cを使って作ってみましょう．

せっかく作るのですからこのFCでは，1バイト単位でファイルを比べていくバイナリ・モードと，行ごとにファイルの違いを見ていくアスキー・モードを選べるようにしましょう．また，大文字，小文字を区別するモードと区別しないモードもあれば便利です．

このように実現したい機能を書き出してコマンドの仕様を決定します．ここでは，FCの仕様を次のようにします．

書式　FC　［スイッチ］　〈ファイル・スペック1〉　〈ファイル・スペック2〉

〈ファイル・スペック1〉で示されるファイルと〈ファイル・スペック2〉で示されるファイルの内容を比べ，違う場所を表示する．

スイッチには次の3つを用意することにする．

/A　アスキー・モードでテキスト・ファイルを比べる．
/B　バイナリ・モードでバイナリ・ファイルを比べる．
/C　英字の大文字，小文字を区別しない．

とくにスイッチを指定しなければ，アスキー・モードで大文字，小文字を区別することにしましょう．

ここまで決まればあとはプログラムを作るだけです．こうしてできたプログラムが**リスト 7.2** です．このプログラムをエディタを使って入力します．

```
#include <stdio.h>
#define     YES 1
#define     NO  0
#define     EVER    ;;

main(argc, argv)
int argc;
char    *argv[];

{
    int i, j, counter = 0;
    int flag1, flag2;
    FILE    *fp1, *fp2;
    char    buf1[255], buf2[255], *fn[2];
    char    caps = NO;
    char    ascii = YES;
    VOID    upstr();

    for(i = 1, j = 0; i < argc; i++){
        if (*argv[i] == '/') {
            switch(toupper(*++argv[i])){
                case 'C':   caps = YES;
                            break;
                case 'B':   ascii = NO;
                            break;
                case 'A':   ascii = YES;
                }
        }else{
            if (j >= 3) {
                printf("too many file names!!");
                exit(1);
            }
```

```
                fn[j++] = argv[i];
            }
        }

        if (ascii) {
            if (((fp1 = fopen(fn[0], "r")) == NULL) ||
                ((fp2 = fopen(fn[1], "r")) == NULL)){
                printf("No file name!!");
                exit(1);
            }

            for(EVER counter++){
                flag1 = (int)fgets(buf1, 255, fp1);
                flag2 = (int)fgets(buf2, 255, fp2);
                if (!(flag1 | flag2)) break;
                if (caps) {
                    upstr(buf1);
                    upstr(buf2);
                }
                if (strcmp(buf1, buf2))
                    printf("line no.%d:¥n%s%s", counter, buf1, buf2);
            }
        }else{
            if (((fp1 = fopen(fn[0], "rb")) == NULL) ||
                ((fp2 = fopen(fn[1], "rb")) == NULL)){
                printf("No file name!");
                exit(1);
            }

            for(EVER counter++){
                flag1 = getc(fp1);
                flag2 = getc(fp2);
                if ((flag1 == EOF) && (flag2 == EOF)) break;
                if ((char)flag1 != (char)flag2)
                    printf("%04x: %02x -- %02x¥n", counter, flag1, flag2);
            }
        }
        printf("complete");
        fclose(fp1);
        fclose(fp2);
}

VOID    upstr(buf)
char    *buf;
{
        for (; *buf != '¥0'; buf++)
            *buf = toupper(*buf);
}
```

リスト 7.2　FC.C

MSX-Cコンパイラのソース・プログラムの拡張子はCでなくてはなりません．MSX-Cは，いくつものプログラムからなり，コンパイルの手順もなかなか複雑なので次のようなバッチ・ファイルを作っておきます．

```
cf -f %1
cg -k %1
m80 =%1/z
del %1.mac
l80 ck,%1,clib/s,crun/s,cend,%1/n/y/e:xmain
```

図7.9 CC.BAT

このバッチ・ファイルを実行すれば，MSX-Cはプログラムをコンパイルし，生成されたアセンブラ・ソースをアセンブルしてCOMファイルにしてくれます．このとき，MSX-CのほかにもアセンブラM80，リンカL80が必要です．いま，これらのプログラムとソース・プログラムであるFC.CはドライブAにあるとします．

必要なファイルがすべてあり，プログラムにもバグがなければ次のような表示になってコンパイル，アセンブル，リンクは終了しCOMファイルができあがります．

```
A>cc fc ⏎
A>cf -f fc ...................................
MSX C ver 1.10P  (parser)                     :
Copyright (C) 1987 by ASCII Corporation       :······ Cコンパイラの起動.
complete                                      :
A>cg -k fc ...................................
MSX C ver 1.10r  (code generator)
Copyright (C) 1987 by ASCII Corporation
main     ..........................................
...............::::::::::::;
upstr    ......:;
complete
```

```
A>m80 =fc/z ……………アセンブラM80の起動.

No Fatal error(s)

A>del fc.mac
A>l80 ck,fc,clib/s,crun/s,cend,fc/n/y/e:xmain …リンカL80の起動.

MSX.L-80  1.00  01-Apr-85  (c) 1981,1985 Microsoft

Data    0103    1E75    < 7538>

35183 Bytes Free
[0103   1E75        30]

A>dir fc.*
FC       C        1584 88-01-12   7:49p ……Cのソース・ファイル
FC       REL      1152 88-01-12   7:35p ……オブジェクト・ファイル
FC       COM      7552 88-01-12   7:36p ……COMファイル
FC       SYM      1152 88-01-12   7:36p ……MSX-S BUGによるデバッグの
                                           際に必要となるファイル.
        4 files   421888 bytes free
A>
```

図7.10　コンパイル

できあがったら，さっそくFCコマンドを使ってみましょう．

```
A>copy con fctest1.doc
yoko
saki
nanno
koihime
^Z
        1 file copied
A>copy con fctest2.doc
Yoko
saki
NANNO
```

サンプルにするテキスト・ファイルを作成.

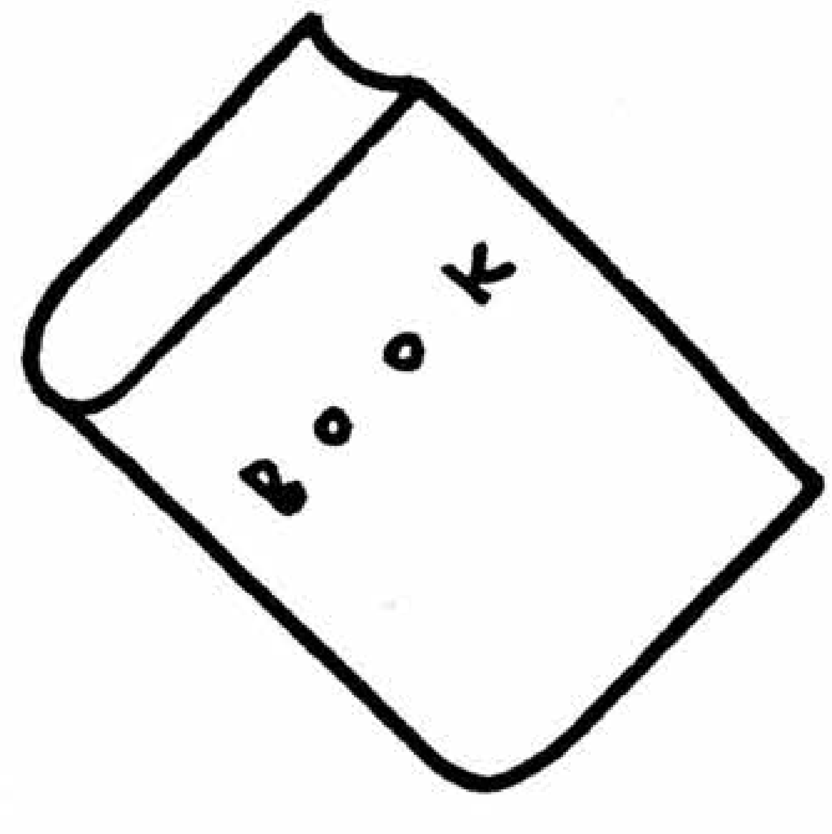
```
asamiya ⏎
^Z ⏎
        1 file copied
A>fc fctest1.doc fctest2.doc ⏎
line no.0:
yoko
Yoko
line no.2:
nanno          オプションを何もしないと,
NANNO          大文字，小文字を区別する.
line no.3:
koihime
asamiya
complete
A>fc /c fctest1.doc fctest2.doc ⏎
line no.3:
KOIHIME        大文字，小文字を区別していない.
ASAMIYA
complete
A>fc /b fctest1.doc fctest2.doc ⏎
0000: 79 -- 59
000C: 6E -- 4E
000D: 61 -- 41
000E: 6E -- 4E
000F: 6E -- 4E
0010: 6F -- 4F
0013: 6B -- 61     バイナリ・モードで比較.
0014: 6F -- 73
0015: 69 -- 61
0016: 68 -- 6D
0018: 6D -- 79
0019: 65 -- 61
complete
A>
```

図 7.11　FC コマンドの実行例

FCは正しく動いているようです．このように，C言語を使えばかなり複雑なコマンドも容易に作ることができます．残念ながらここではC言語のプログラムの具体的な説明はできませんが，DOSに密着したプログラムが非常に

書きやすい言語であるということは覚えておいてください．

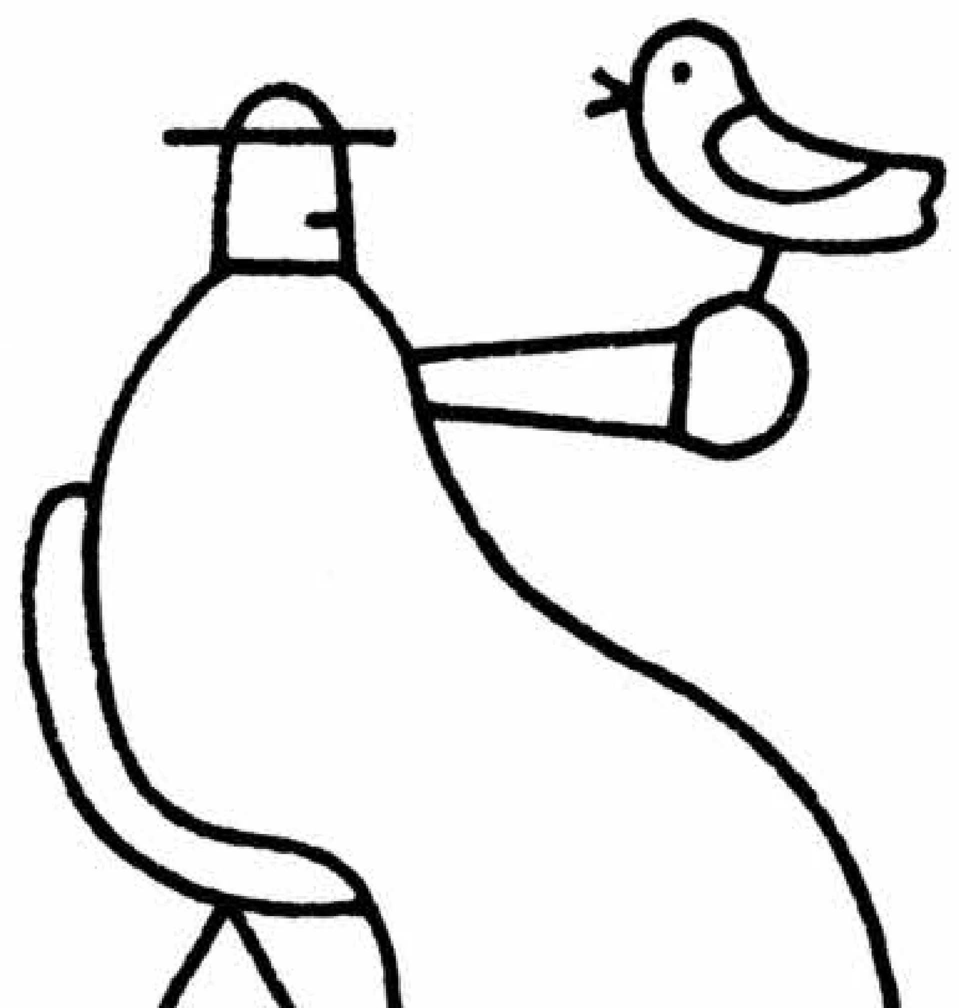
もちろん，MSX-DOS 上での高級言語は C 言語だけではありません．DOS のユーティリティを作るのに適した C 言語のように，人工知能のプログラミングに適した LISP，数値計算に適した FORTRAN，構造化プログラミング言語の PASCAL など，さまざまな言語が用意されています．アセンブリ言語も忘れてはなりません．このほかにも，DOS 上にはさまざまな言語があります．目的に応じて最も適した言語を使い分けるのが肝心です．

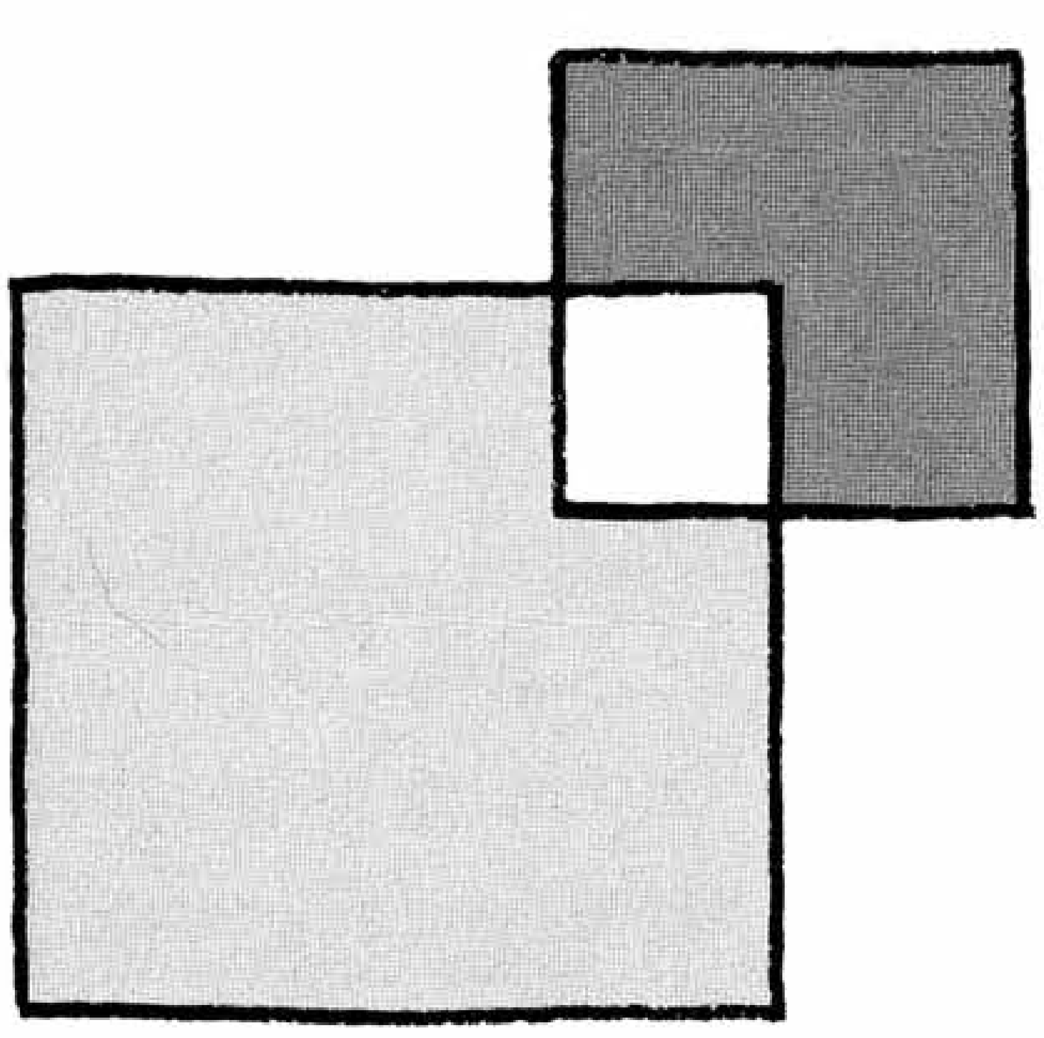


8章

MSX-DOSと MS-DOSとの関係

― ファイルの互換性について ―





8-1 ファイルの互換性

前章までにも何回か，MSX-DOSの特徴としてMS-DOSとファイルの互換性があるということを述べました．この章では，〝MS-DOSとファイルの互換性がある〞というのは，具体的にはどのようなことを指しているのかを説明していきます．また，本章でMS-DOSを使う例がいくつかありますが，これらはMSXでは実験できませんので，図だけ見てもらえば十分です．将来MS-DOSを使う機会があったら再び参照してください．

MS-DOSのディスク・フォーマットにはいくつか種類がありますが，MSX-DOSと同じ2DDの8セクタと9セクタ・フォーマットもあります(フォーマットの種類，セクタについては2章のコラム「MSX-DOSのディスク・フォーマット」を参照)．MSX-DOSとの互換性を考えるときは，8セクタフォーマットを使えないMSXもあるので，9セクタフォーマットを使うようにします．

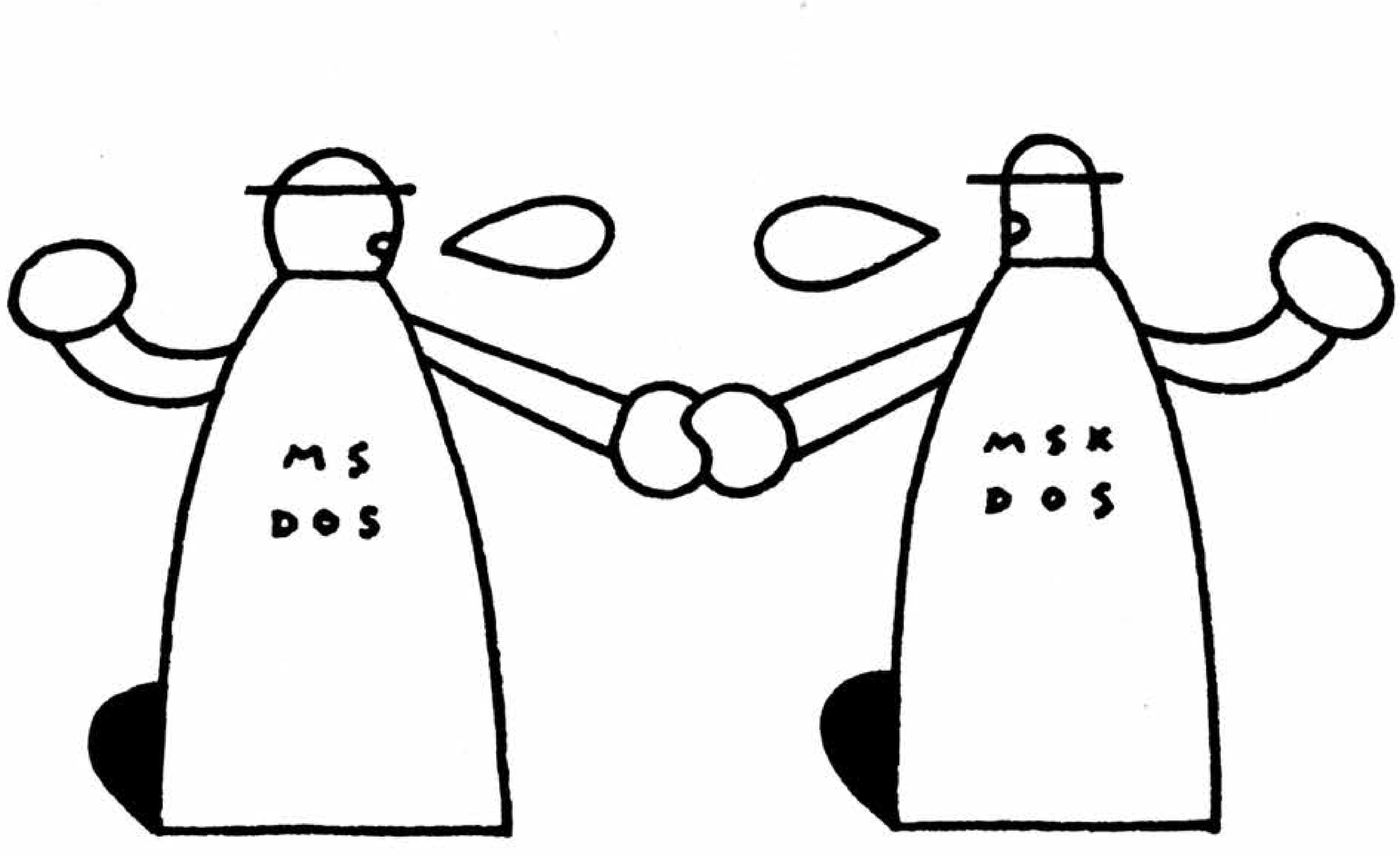


図8.1を見てください.

```
A>dir ⏎ ……………………… MS-DOS上でMSX-DOS TOOLSの
                          ディレクトリを表示させる.

 ドライブ A: のディスクのボリュームラベルはありません.
 ディレクトリは A:¥

MSXDOS   SYS     2432  85-08-23  21:29
COMMAND  COM     6656  85-09-02  22:10
AUTOEXEC BAT       77  87-02-12   3:58
M80      COM    20480  85-07-11  18:46
L80      COM    10752  85-07-11  19:01
CRE                    85-04-01  22:22
VIEW     COM     9600
WC       COM     9216  87-03-04
WELCOME  DOC      372  87-02-12  15:01
SAMPLE1  DOC       25  87-02-11  20:47
SAMPLE2  DOC       97  87-02-11  20:47
SAMPLE3  DOC       61  87-02-11  20:48
SEARCH   BAT      128  87-01-16  19:28
SEARCH   MAC     2048  87-01-16  19:28
SEARCH   BAS      357  87-02-12   6:41
        45 個のファイルがあります.
    372736 バイトが使用可能です.

A>
```

図8.1　MS-DOS上でのディレクトリ表示(MS-DOS)

これは, MS-DOSでMSX-DOSのディスクのディレクトリを表示させているところです. プロンプトといい, 表示される内容といい, MSX-DOSとほとんど同じです.

このように, MSX-DOSのディスクのディレクトリをMS-DOSで表示させることができます. もちろん, 反対にMS-DOSのディスクのディレクトリをMSX-DOSで表示させることも可能です(ただし, 漢字のファイル名は表示できません).

MS-DOSで即実行可能なファイルの拡張子にはCOMとEXEという2

種類があります．これらの区別はMS-DOS特有のものですのでここでは説明しませんが，とにかくMS-DOSにもCOMファイルがあります．しかし，MSX-DOSのCOMファイルをMS-DOSで実行することはできません．MSXは8ビットのCPUを使っていますが，MS-DOSは16ビットのCPUを使っているために，両者のマシン語はまったく違います．このため同じことをするプログラムを書いても，マシン語であるCOMファイルの中身はまるっきり違っているのです．

では，MSX-DOSとMS-DOSの互換性は単にディレクトリをお互いに表示できるというだけなのでしょうか？　いいえ，そんなことはありません．マシン語が違っていてもテキスト・ファイルは共通に使えます．MSXで作ったテキスト・ファイルをMS-DOSで読んでみましょう．MSX-DOSに組み込まれているコマンドは，MS-DOSでもほとんど使えます(もちろん，さらに高機能なコマンドも用意されています)．ここでは，テキスト・ファイルの内容を表示させるのですからTYPEコマンドを使いましょう．

```
A>type message.doc ⏎ ………… MSX-DOS上で作成したテキスト・ファイルを
Good morning, Dr Chandra!     MS-DOSに読み込ませ，表示させる．

I'm HAL-9000 Computer.

A>
```

図8.2　MSX-DOSで作成したテキスト・ファイルをMS-DOS上で表示（MS-DOS）

図8.2のように，テキスト・ファイルはきちんと読むことができます．このことは大きな意味を持ちます．MSX-DOSのCOMファイルをMS-DOS上で直接実行させることはできませんが，7章でも述べたように，COMファイルも最初からマシン語だったわけではありません．アセンブラで作られたものはどうにもなりませんが，高級言語で作られたものならば，そのソース・プログラム(コンパイルするまえのプログラム，テキスト・ファイルになっている)をMS-DOSにもってきて，MS-DOS上のコンパイラでコンパイルすれば，MSX-DOSのプログラムをMS-DOS上で動かすことが可能なのです．

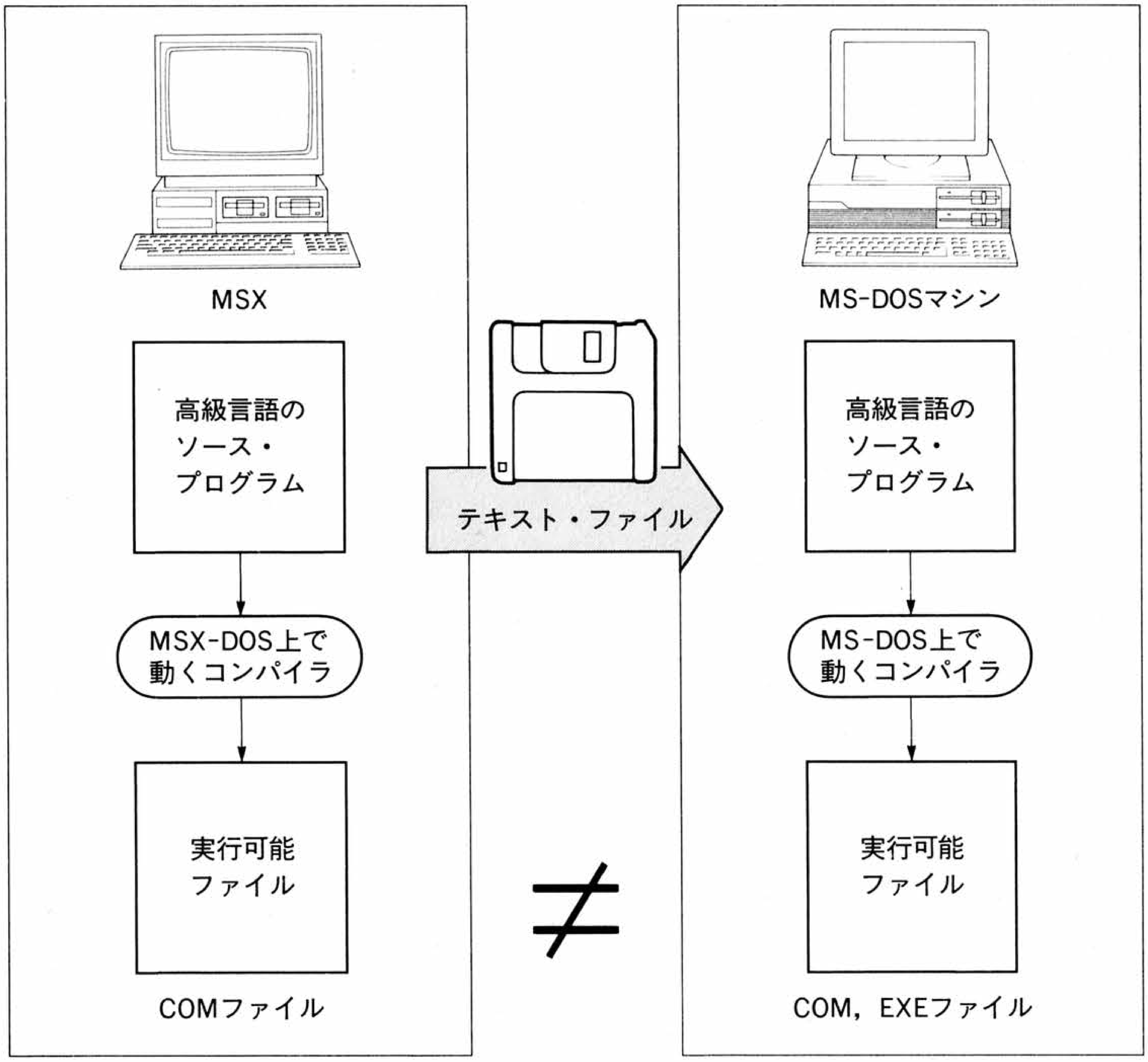


図8.3　MSX-DOSのプログラムをMS-DOS上で動かす

ただし，すべてのテキスト・ファイルがMS-DOSとMSX-DOSで共通に扱えるわけではありません．7章でECHOを作った際に説明しましたが，カーソルなどを制御する特別な文字にESCシーケンスというものがありました．このESCシーケンスが，MS-DOSではVT-100コンパチと呼ばれる規格なのですが，一方MSX-DOSでは**VT-52サブセット**という規格(表7.2参照)なので，この点については互換性がありません．ESCシーケンスを含むテキスト・ファイルをMS-DOSとMSX-DOSの間でやりとりをする場合には，この点に十分気を配る必要があります．

8 2 ファイルの属性

MS-DOSのファイルの中にはさまざまな属性を持つものがあります．たとえば，書き込みが禁止された読み出し専用ファイルとか，ディレクトリを表示させても表示されない隠しファイルなどです．図8.4に例を示します．

```
A>chkdsk /v ⏎ ……………ディスクの中身をチェックするMS-DOSのコマンド.
                       MSX-DOS TOOLSにも同様のものがある.
ディレクトリ A:¥
      A:¥IO.SYS     } 隠しファイル
      A:¥MSDOS.SYS  }
      A:¥COMMAND.COM ……………とくに属性は持たない
      A:¥READONLY.DOC ……………読み出し専用ファイル
      A:¥HIDDEN.DOC ……………隠しファイル
ディレクトリ A:¥SUB-DIR ………サブ・ディレクトリ. この中にファイルを
                              作ることができる.

   730112 バイト : 全ディスク容量
    62464 バイト : 3 個のシステムファイル
     1024 バイト : 1 個のディレクトリ
    25600 バイト : 2 個のユーザーファイル
   641024 バイト : 使用可能ディスク容量

   655360 バイト : 全メモリ
   416624 バイト : 使用可能メモリ

 ドライブ A: のディスクのボリュームラベルはありません.
 ディレクトリは A:¥

A>dir ⏎

COMMAND  COM     24145  86-10-13   0:00 } 隠しファイルのディレクトリ
READONLY DOC        25  88-01-11  21:28 } は表示されない.
SUB-DIR      <DIR>      88-01-11  22:11 }
```

```
        3 個のファイルがあります.
   641024 バイトが使用可能です.

A>del readonly.doc ⏎
アクセスは拒否されました.
```

……読み出し専用ファイルは，書き込みや削除ができない．

図 8.4　MS-DOS のファイル属性（MS-DOS）

これらのファイルを MSX-DOS で見てみると，どのように見えるでしょうか？　実際に MSX-DOS でこのディスクのディレクトリを表示させてみると図 8.5 のようになります．

```
A>dir ⏎
COMMAND  COM    24145 86-10-13 12:00a
READONLY DOC       25 88-01-11  9:28p
SUB-DIR        <DIR>  88-01-11 10:11p
        3 files   641024 bytes free
A>del readonly.doc ⏎
A>dir ⏎
COMMAND  COM    24145 86-10-13 12:00a
SUB-DIR        <DIR>  88-01-11 10:11p
        2 files   642048 bytes free
A>
```

MS-DOSの各属性を持つファイルのはいったディスクをMSX-DOS上に持ってきて，ディレクトリを表示させる．隠しファイルは，やはり表示されない．

MS-DOS上で削除することのできできなかったファイルが，MSX-DOS上では削除できてしまう．

図 8.5　MSX-DOS 上でのファイル属性（MSX-DOS）

見たところ，とくに変わりありません．ところが，MSX-DOS でこれらのファイルをアクセスするとなると，話は変わります．実は MSX-DOS では，ファイルに属性を持たせることはできません．読み出し専用ファイルは普通のファイルとして認識され属性は無視されますが，その他の属性を持ったファイルは MSX-DOS ではファイルとして認識されなくなってしまいます．

つまり，読み出し専用ファイルは，MSX-DOSでは，普通のファイルとして読み書きができてしまい，その他の属性を持ったファイルに対しては，あらゆるアクセスが不可能となってしまいます．もちろん，MSX-DOSではサブ･ディレクトリにアクセスすることもできません．

MS-DOSとMSX-DOSで互いに読み書きするファイルは，普通はテキスト･ファイルだけです．ですから何かの属性が付いているとは考えられませんが，もし属性が付いていると，MSX-DOSで正常にファイルとしてアクセスできませんので，かならずMS-DOS側で属性を変更してからMSX-DOSでアクセスしなくてはなりません．**図8.6**はMS-DOS上で隠しファイルを普通のファイルに直している例です．

```
A>chmod -h hidden.doc    ……………ファイルの属性を変更する
                                  MS-DOSのコマンド．
A>dir

 ドライブ A: のディスクのボリュームラベルはありません．
 ディレクトリは A:¥

COMMAND  COM     24145  86-10-13   0:00    これでMSX-DOS上でも読み
HIDDEN   DOC        22  88-01-11  21:29……込むことができる．
SUB-DIR      <DIR>      88-01-11  22:11…サブ・ディレクトリはMSX-DOS
         3 個のファイルがあります．          では絶対にアクセスすることは
   642048 バイトが使用可能です．            できない．

A>
```

図8.6　属性の変更例(MS-DOS)

図8.6の隠しファイルHIDDEN.DOCは属性を変更して普通のファイルになったので，MSX-DOS上で読むことができます．実際にやってみましょう．**図8.7**を見るとわかるように，今度はディレクトリも表示でき，TYPEコマンドで内容を表示することもできます．

```
A>dir ⏎
COMMAND  COM    24145 86-10-13 12:00a
HIDDEN   DOC       22 88-01-11  9:29p ……ディレクトリが表示できている.
SUB-DIR       <DIR>   88-01-11 10:11p
       3 files   642048 bytes free
A>type hidden.doc ⏎
This is hidden file. ……………ファイルが読めている.

A>
```

図8.7　属性変更後のファイルをMSX-DOSで読む（MSX-DOS）

本書では，アプリケーション・ソフトウェアなどを使うために必要なMSX-DOSの知識を説明してきました．ここまで読み進んできた皆さんはすでにDOSの基本的知識を身に付けているはずです．

しかし，言語の具体的な説明や，アプリケーション・ソフトウェアの使い方についてはほとんど本書では触れていません．これらは，皆さんが各自の興味に応じて学んでいってください．

MSX-DOSの世界はこれで終わりではありません．皆さんの工夫次第で無限の可能性を秘めています．本書が皆さんとDOSの世界の橋渡しとなれば，これほどうれしいことはありません．

MSX-DOSの世界はここから始まるのです．

Appendix

MSX-DOS コマンド一覧

[] 内は省略可.

● BASIC　BASIC の起動

書式　BASIC　[〈ファイル・スペック〉]

DISK-BASIC を起動する．起動時に〈ファイル・スペック〉のファイルを読み込む．

● COPY　ファイルのコピー，連結

書式　COPY　〈ファイル・スペック 1〉　〈ファイル・スペック 2〉

〈ファイル・スペック 1〉のファイルを〈ファイル・スペック 2〉にコピーする．スイッチによりモードを選択可能．

書式　COPY　〈ファイル・スペック 1〉 ＋ 〈ファイル・スペック 2〉 ＋…
… ＋ 〈ファイル・スペック n〉　〈ファイル・スペック〉

〈ファイル・スペック 1〉～〈ファイル・スペック n〉までのファイルを連結し，〈ファイル・スペック〉にコピーする．スイッチによりモードを選択可能．

／A　アスキー・モードでコピー，あるいは連結
／B　バイナリ・モードでコピー，あるいは連結

● DATE　日付の設定

書式　DATE　[〈年〉－〈月〉－〈日〉]

内蔵カレンダの日付を設定する．引数を省略した場合には，現在の日付を表示し，新しい日付の入力を求めてくる．〝－〟は〝／〟でも可．

● DEL(ERASE)　ファイルの削除

書式　DEL　〈ファイル・スペック〉
　　　　ERASE　〈ファイル・スペック〉

〈ファイル・スペック〉で指定したファイルをディスク上から削除する．

● DIR　ディレクトリの表示

書式　DIR　[〈ファイル・スペック〉]

〈ファイル・スペック〉で指定したファイルのディレクトリを表示する．何も指定しない場合は，カレント・ドライブのすべてのファイルについてのディレクトリを表示する．スイッチによりモードを選択可能．

／P　ページ・モードで表示
／W　ワイド・ディスプレイ・モードで表示

● FORMAT　ディスクのフォーマット

書式　FORMAT

コマンド実行後に表示されるメニューの選択によりディスクを初期化．

● MODE　表示文字幅の設定

書式　MODE　〈表示文字幅〉

ディスプレイへの表示文字幅を設定する．〈表示文字幅〉は MSX の場合 1～40，MSX2 の場合は 1～80 を指定する．

● PAUSE　バッチ・ファイルの実行一時停止とメッセージ出力

書式　PAUSE　[〈メッセージ〉]

バッチ・ファイル中にこのコマンドがあればそこで実行を一時停止し，〈メッセージ〉があればそれを表示する．

● REM　バッチ・ファイル中のコメント

書式　REM　[〈コメント〉]

バッチ・ファイル中にコメントを入れる．ただし，〈コメント〉はバッチ・ファイル実行時に画面に表示される．

● REN(RENAME)　ファイル名の変更

書式　REN　〈ファイル・スペック〉　〈ファイル名〉

〈ファイル・スペック〉で指定したファイルのファイル名を〈ファイル名〉に変更する

● TIME　時刻の設定

書式　TIME　[〈時〉：〈分〉：〈秒〉]

内蔵時計の時刻を設定する．引数を省略した場合には，現在の時刻を表示し，新しい時刻の入力を求めてくる．

● TYPE　テキスト・ファイルの内容を画面に表示

書式　TYPE　〈ファイル・スペック〉

〈ファイル・スペック〉で指定したテキスト・ファイルを画面に表示する．

● VERIFY　ディスクへ書き込む際のチェック

書式　VERIFY　{ON／OFF}

ディスクにデータを書き込む際に，正しく書き込まれているかどうかチェックを行う．

エラー・メッセージ一覧

● Bad command or file name
コマンド，またはファイル名が違います．

● Bad FAT, drive 〈ドライブ名〉
ドライブ〈ドライブ名〉のディスクのFATが破壊されています．

● Bad parameter
引数が違います．(FORMAT コマンド実行時)

● Boot error
MSX-DOS を起動することができません．

● Disk error
ディスクにエラーが発生しました．(FORMAT コマンド実行時)

- **Disk error reading drive 〈ドライブ名〉**
  Disk error writing drive 〈ドライブ名〉
  ドライブ〈ドライブ名〉のディスクにエラーが発生しました.
- **File cannot be copied onto itself**
  ファイルをそれ自身にはコピーできません.
- **File creation error**
  ファイルを作れません.
- **File not found**
  ファイルが見つかりません.
- **Insert disk with batch file**
  バッチ・ファイルのあるディスクをドライブに入れてください.
- **Insert diskette for drive 〈ドライブ名〉**
  ディスクをドライブ〈ドライブ名〉に入れてください.
- **Insert DOS disk in default drive**
  COMMAND.COM のはいったディスクをデフォルト・ドライブ(カレント・ドライブ)に入れてください.
- **Insufficient disk space**
  ディスクの容量が足りません.
- **Invalid date**
  日付の指定が違います.
- **Invalid drive specification**
  ドライブの指定が違います.
- **Invalid parameter**
  引数が違います.(MODE コマンド, VERIFY コマンド実行時)
- **Invalid time**
  時刻の指定が違います.
- **No enough memory**
  メモリが足りません.
- **Not ready error reading drive 〈ドライブ名〉**
  ドライブ〈ドライブ名〉は読み込みの準備ができていません.

● **Not ready error writing drive 〈ドライブ名〉**
ドライブ〈ドライブ名〉は書き込みの準備ができていません.

● **Program too big to fit in memory**
プログラムが大き過ぎて，メモリに収まりません.

● **Rename error**
このファイル名変更は間違っています.

● **Unsupported media type error reading drive 〈ドライブ名〉**
ドライブ〈ドライブ名〉のディスクは，このドライブでは扱えません.

● **Write error**
書き込みができません.

● **Write protect error reading drive 〈ドライブ名〉**
ドライブ〈ドライブ名〉のディスクからの読み込み中に，ライト・プロテクト・エラーが生じました.

● **Write protect error writing drive 〈ドライブ名〉**
ドライブ〈ドライブ名〉のディスクに対して書き込みができません.

入力待ちメッセージ一覧

以下のメッセージは，ユーザーに入力を求めるものであり，エラーメッセージなどと組み合わせて使われることもあります.

● **Abort, Retory, Ignore?**
中止(A)，再試行(R)，無視(I)？

● **Are you sure (Y／N)?**
よろしいですか(Y／N)？

● **Enter new date:**
日付を入力してください.

● **Enter new time:**
時刻を入力してください.

● **Press any key for retry**
任意のキーを押し，再試行してください.

- Strike any key when ready
  準備ができたら，任意のキーを押してください．

- Terminate batch file (Y／N)?
  バッチ処理を中止しますか(Y／N)?

本書で紹介したソフトウェア

- 「HALNOTE」　ユーザー・インターフェイス，日本語ワードプロセッサ，図形プロセッサ

価　　格　¥29,800
発 売 元　株式会社 HAL 研究所
問い合わせ先　TEL 03-252-5561

- 「MSX-C」　C コンパイラ

価　　格　¥19,800

- 「MSX-DOS TOOLS」　MSX-DOS システム，各種ユーティリティほか

価　　格　¥14,800

- 「MSX-S BUG」　シンボリック・デバッガ

価　　格　¥19,800

- 「日本語 MSX-Write」　日本語ワードプロッセッサ

価　　格　¥19,800

以上4点の発売元　株式会社アスキー
問い合わせ先　TEL 03-486-8080(営業技術部)

本書の執筆にあたり，ソフトウェアの提供に快く応じてくださった株式会社HAL研究所，並びに株式会社アスキーの両社にお礼申し上げます．

著者

索　引

A

C

D

E

F

G

H

I

J

K

L

M

参考文献

・「MSX-DOS MANUAL」
・「MSX2　テクニカル・ハンドブック」
・「MSX2 BASIC 入門」
・「入門 MS-DOS」
・「標準 MS-DOS ハンドブック」

以上，株式会社アスキー

MSX-DOS 入門

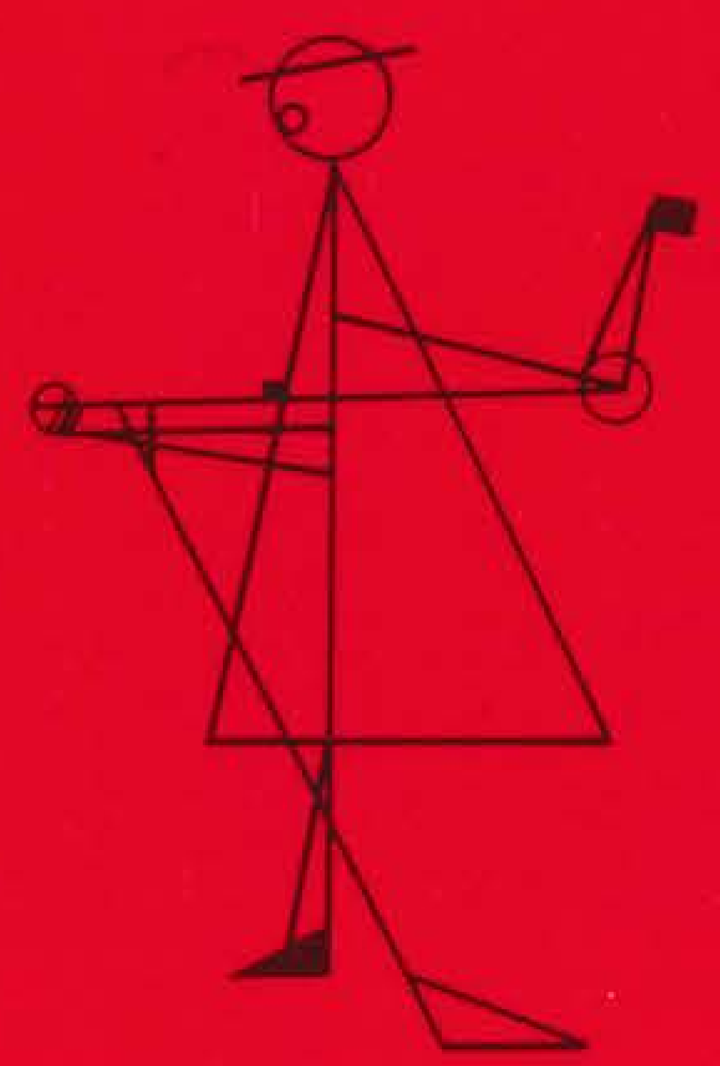
1988年3月1日　初版発行
定価1,200円

著　者　中村　哲（なかむら　さとし）
発行者　塚本慶一郎
発行所　株式会社アスキー
〒107-24　東京都港区南青山6-11-1スリーエフ南青山ビル
振　替　東京4-161144
TEL（03）486-7111（大代表）
情報 TEL（03）498-0299（ダイヤルイン）
出版営業部 TEL（03）486-1977（ダイヤルイン）

制　作　株式会社GARO
ＣＴＳ／印刷　壮光舎印刷株式会社

編集　土屋信明・竹内充彦
本文デザイン　川戸明子

ISBN4-87148-295-2 C3055 ¥1200E



ISBN4-87148-295-2 C3055 ¥1200E

定価 1,200円